NULL-SPACE METHODS FOR NUMERICAL SOLUTIONS OF DIFFERENTIAL EQUATIONS

BY

HANNA JOY NERADT

B.A., Trinity Christian College, 2000

DISSERTATION

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Computer Science
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2007

Urbana, Illinois

Abstract

A linear differential operator often has a nontrivial null space. One consequence for such an operator, $\mathcal{L}$, is that solutions of equations of the form $\mathcal{L}u = f$ are never unique if the null space contains more than the zero element: adding any nontrivial null function to any particular solution yields another solution. Out of the infinite set of functions satisfying the differential equation, the conventional way to select a desired solution is to require the solution (or its derivatives) to take on prescribed values at specified points, typically on the boundary of the domain over which the equation is defined. For some applications, however, it may be inconvenient or inappropriate to specify such boundary conditions. For certain problems in materials science, for example, it is more natural to specify the null-space component of the solution directly rather than indirectly via (often unknown) boundary conditions.

One particular example is atomic scale simulation of stress in metals. When calculations are made on a tiny scale, the edges of the metal, which would provide a boundary for the problem, are practically an infinite distance away. In this case, the conventional method is difficult to apply, even if it were computationally feasible. Instead, a natural alternative is to specify the null-space component to single out a particular solution.

In this thesis, we will develop numerical methods for computing approximate solutions to linear differential equations subject to explicit specification of the null-space component. For this purpose we will develop discretized approximations to the null spaces of relevant differential operators as well as numerical solution procedures that take advantage of such an explicit representation. To the best of our knowledge, this explicit null-space approach and our implementation of it are new.

This thesis details the problem we seek to solve, discusses options for finding null bases, explains the explicit null bases we have found, and demonstrates a solution technique for solving the problem referenced above using a null space method.

To:

- *Brian–thanks for trusting God would provide for me to finish this work*
- *Mom and Dad Vander Zee–thanks for the many prayers on my behalf*
- *Mom and Dad Neradt, my "other" parents–thanks for accepting me into your family*
- *my siblings–you each hold a special place in my heart*

Acknowledgements

This research was supported by the endowment for the Fulton Watson Copp Chair in Computer Science held by my advisor. Further funding was provided by the University of Illinois at Urbana-Champaign through a Scholarship for Under-Represented Groups in Engineering and through an Illiac Scholarship.

The author would like to thank Professor Michael Heath for his guidance, insights, and encouragement throughout the course of this research. Further, the author acknowledges Yahweh, the God of the Israelites, revealed through Jesus Christ, my Savior, as the author of all knowledge and understanding.

Table of Contents

List of Figures

1 Introduction and Motivation

A linear differential operator often has a nontrivial null space. One consequence for such an operator, $\mathcal{L}$, is that solutions of equations of the form $\mathcal{L}u = f$ are never unique if the null space contains more than the zero element: adding any nontrivial null function to any particular solution yields another solution. Out of the infinite set of functions satisfying the differential equation, the conventional way to select a desired solution is to require the solution (or its derivatives) to take on prescribed values at specified points, typically on the boundary of the domain over which the equation is defined. For some applications, however, it may be inconvenient or inappropriate to specify such boundary conditions. For certain problems in materials science, for example, it is more natural to specify the null-space component of the solution directly rather than indirectly via (often unknown) boundary conditions.

One particular example is atomic scale simulation of stress in metals [2]. When calculations are made on a tiny scale, the edges of the metal, which would provide a boundary for the problem, are practically an infinite distance away. In this case, the conventional method is difficult to apply, even if it were computationally feasible. Instead, a natural alternative is to specify the null-space component to single out a particular solution. This application is the motivation for our research into computationally viable null-space methods. In this thesis, we will develop numerical methods for computing approximate solutions to linear differential equations subject to explicit specification of the null-space component. This null-space approach is generally applicable to differential equations without boundary specifications and to problems with boundaries at infinity, such as the problem described above, the Poisson equation in electrostatics, and Maxwell's equations in electromagnetics. While the methods are not yet robust for large-scale simulations, we lay a foundation for these methods, we begin to address the issues that arise in this new approach to solving differential equations, and we demonstrate the accuracy of this approach by solving several example and real-life problems. For simplicity, the specific problems we consider are discretized uniformly and have the same number of nodes in each dimension.

To the best of our knowledge, this explicit null-space approach and our implementation of it are new.

This thesis details the problem we seek to solve, discusses options for finding null bases, explains the explicit null bases we have found, and demonstrates a

solution technique for solving the motivating problem referenced above.

Chapters 2-3 give background information on linear spaces and linear systems, including the somewhat novel notation we developed for this research that will be used throughout the thesis. We explain the general approach we take to solving differential equations with null-space specifications in Chapter 4 and provide an overview of discretization options in Chapter 5. Together, these two chapters give the nuts and bolts of the null-space approach, including a proof that this technique produces a linear system with a unique solution. They provide the groundwork for our research in proving the validity of this approach and in explaining problem formulations. Chapter 5 includes convergence results for some two-dimensional problems.

In Chapter 6 we deal with the subproblem of finding a discrete basis for the null space, specifically when the discrete operator is sparse. We review previous research on the topic, explain and validate explicit, discrete approximations to null spaces of some relevant differential operators, and present heuristics for improving conditioning of discrete bases without sacrificing much sparsity. We put special emphasis on this subproblem because the task of finding a null-space basis may dominate the overall computational complexity of null-space methods and because little research has been done addressing the trade-off between sparsity and good conditioning. Our explicit bases easily satisfy the sparsity and computational speed desired for null-space methods, but some of the bases are ill-conditioned. In order to address the ill-conditioning, we developed heuristics that sacrifice some sparsity for improved conditioning. The end of the chapter includes various data about our explicit bases and other general methods for finding a null-space basis, as well as information about the performance of our conditioning heuristics for several matrices.

In Chapter 7, we discuss techniques for solving the discrete linear system, giving several approaches to solving the system and reviewing preconditioning options. We also provide computational results for some example problems. Specific applications are presented in Chapter 8, and we summarize our work and explain possible continuations and extensions of this research in Chapter 9.

Throughout the text, we use shortened versions of phrases to save space and avoid wordiness. The most common short-hand phrases are defined here.

- null basis: a basis for a null space
- null function: a function in the null space of a continuous operator
- null vector: a vector in the null space of a discrete operator
- problem of size n: problem with a uniformly discretized domain, having n nodes in each spatial dimension

We also include here a list of symbols we often use and the object each represents.

- $\mathcal{D}$, differential operator
- $\boldsymbol{D}$, discretized differential operator
- u, solution function
- $\boldsymbol{u}$, discretized solution
- v, null function
- ϕ, basis function

2 Linear Spaces

In this chapter we review background material and establish terminology and notation that will be needed later. Most of what we say here is applicable to linear spaces in general, but we couch much of the discussion in terms of function spaces, since that will be our main application. Our applications will require only the real scalar field $\mathbb{R}$, but essentially the same development is equally valid with the complex scalar field, $\mathbb{C}$. Many topics in this chapter are discussed more thoroughly in [35, ch. 4].

2.1 Function Spaces

This thesis is concerned with numerical approximations associated with discretizations of linear differential operators. Differential operators operate on functions, so the natural habitats in which to study them are *function spaces*. The set of all real-valued functions defined on some domain Ω in d-dimensional Euclidean space $\mathbb{R}^d$, with addition and scalar multiplication defined pointwise (i.e., $(f+g)(t) = f(t)+g(t)$ and $(\alpha f)(t) = \alpha f(t)$ for $t \in \Omega$), form a *linear space* or *vector space* satisfying the usual properties of vector addition and scalar multiplication. The zero vector for such a space is the constant function that is identically zero on the domain Ω.

Perhaps the most familiar example of such a function space, though we may not often think of it that way, is to take $\Omega = \{1, 2, \ldots, n\}$ for some positive integer n, in which case the resulting linear space is simply $\mathbb{R}^n$, where for $\boldsymbol{x} \in \mathbb{R}^n$ we normally use component notation x_i instead of function notation $x(i)$, $i = 1, \ldots, n$. Similarly, the linear space obtained by taking Ω to be integer lattice points (or scalar multiples thereof) in $\mathbb{R}^2$ can be identified with $\mathbb{R}^{n\times n}$, etc. Such discrete domains Ω are the natural home for *difference operators* rather than differential operators, as the latter operate on smooth functions of continuous variables. Difference operators will nevertheless be of interest to us as approximations to differential operators.

Since derivatives of a function are properly defined only at interior points of its domain, the domain Ω for a differential equation must have nonempty interior. Though the results we will develop apply to any such domain with reasonably regular boundary, for illustrative purposes we will typically take Ω to be the unit interval $(0,1) \subseteq \mathbb{R}$, the unit square $(0,1)\times(0,1) \subseteq \mathbb{R}^2$, or the unit cube $(0,1)\times(0,1)\times(0,1) \subseteq \mathbb{R}^3$, depending on the desired dimension. In

any case, we will denote the boundary of Ω by $\partial\Omega$.

To yield a useful structure, the functions defined on Ω must also possess some degree of regularity. The functions must be suitably smooth in order for a given differential operator to be applicable, but for reasons that will become apparent later, we will take our fundamental linear space to be $L_2(\Omega)$, the space of all real-valued, square-integrable functions defined on some domain Ω, which we take to be an open set in $\mathbb{R}^d$ with a piecewise smooth boundary.

One reason for focusing on $L_2(\Omega)$ is that we can define the *inner product* for any functions $f, g \in L_2(\Omega)$ as the integral of their pointwise product,

$$\langle f, g \rangle = \int_\Omega fg, \tag{2.1}$$

and in turn define the *norm* of any function $f \in L_2(\Omega)$ by

$$\|f\| = \langle f, f \rangle^{1/2} = \left(\int_\Omega f^2 \right)^{1/2}. \tag{2.2}$$

So equipped, $L_2(\Omega)$ is both an *inner product space* and a *normed linear space*. Further, for the Lebesgue integral it can be shown that every Cauchy sequence in $L_2(\Omega)$ converges, so $L_2(\Omega)$ is in fact both a *Hilbert space* (i.e., a complete inner product space) and a *Banach space* (i.e., a complete normed linear space).

2.2 Subspaces and Bases

We will frequently be interested in finite linear combinations of functions, say $\{v_i\}_{i=1}^n \subseteq \mathcal{F}(\Omega)$, where $\mathcal{F}(\Omega)$ is a function space defined on a domain Ω, so we adopt the following matrix-like notation to describe such combinations compactly. (While this notation is uncommon in our experience, similar notation, likewise associated with functions as columns of matrices, appears in [5, 18, 50, 53].) Thinking of the functions as column vectors (even when the cardinality of Ω is not finite or even countable), we write the corresponding matrix as $\boldsymbol{V} = [\, v_1 \quad v_2 \quad \cdots \quad v_n \,]$, and for $\boldsymbol{x} \in \mathbb{R}^n$ we define the product $\boldsymbol{V}\boldsymbol{x} \in \mathcal{F}(\Omega)$ by

$$\boldsymbol{V}\boldsymbol{x} = \sum_{i=1}^n x_i v_i, \tag{2.3}$$

i.e., the linear combination of columns (functions) of $\boldsymbol{V}$ determined by the n scalar entries of the vector $\boldsymbol{x}$. More generally, if $\boldsymbol{X} \in \mathbb{R}^{n \times k}$ is an $n \times k$ matrix, then

$$\boldsymbol{V}\boldsymbol{X} = \left[\begin{array}{ccc} \sum_{i=1}^n x_{i,1} v_i & \cdots & \sum_{i=1}^n x_{i,k} v_i \end{array} \right] \tag{2.4}$$

is a matrix, each of whose k columns is a function in $\mathcal{F}(\Omega)$. Note that this agrees with conventional matrix notation when Ω is finite; when Ω is infinite, we can informally think of the row dimension of the corresponding matrix as

being infinite. Henceforth, we will identify a finite set of functions $\{v_i\}_{i=1}^n$ with the matrix $\boldsymbol{V}$ composed of those functions.

A set of functions is said to be *linearly dependent* if it contains a finite subset $\boldsymbol{V}$ such that $\boldsymbol{V}\boldsymbol{x} = 0$ for some $\boldsymbol{x} \neq \boldsymbol{0}$, i.e., a nontrivial finite linear combination of the functions vanishes identically on Ω, and otherwise is said to be *linearly independent*. For example, the monomials $v_i(t) = t^{i-1}$, $i = 1, \ldots$ are linearly independent on $(0, 1)$. The subset

$$\text{span}(\boldsymbol{V}) = \{\boldsymbol{V}\boldsymbol{x} : \boldsymbol{x} \in \mathbb{R}^n\} \subseteq \mathcal{F}(\Omega) \tag{2.5}$$

of all linear combinations of a set of functions $\boldsymbol{V}$ is itself a linear space, called the *span* of $\boldsymbol{V}$. For example, the monomials $v_i(t) = t^{i-1}$, $i = 1, \ldots, n$, span the linear space $\mathbb{P}_{n-1} \subseteq L_2(0, 1)$ of polynomials on $(0, 1)$ of degree at most $n - 1$. Such a subset that is itself a linear space is called a *subspace* of the larger linear space containing it. As another example, the set of all functions in $L_2(\Omega)$ that vanish on a given fixed subset of $\bar{\Omega}$, such as $\partial\Omega$, form a subspace of $L_2(\Omega)$; successively larger subsets would yield successively smaller such subspaces.

Note that a subspace may or may not be *closed*, i.e., it may or may not contain all of its limit points. For example, for any finite set of functions $\boldsymbol{V} \subseteq L_2(\Omega)$, $\text{span}(\boldsymbol{V})$ is a closed subspace, but the subspace $C(\Omega)$ of all *continuous* real-valued functions on Ω is not closed, since the limit (in $L_2(\Omega)$) of a sequence of continuous functions is not necessarily continuous. Indeed, $L_2(\Omega)$ contains a hierarchy of successively smaller subspaces composed of successively smoother functions (continuous, Lipschitz continuous, differentiable, continuously differentiable, twice-differentiable, etc.) that are dense in $L_2(\Omega)$, but none is closed.

A *basis* for a function space $\mathcal{F}(\Omega)$ is a set of linearly independent functions $\boldsymbol{V}$ such that $\text{span}(\boldsymbol{V}) = \mathcal{F}(\Omega)$. Assuming the Axiom of Choice, every linear space has a basis, but that basis need not be finite. For example, $L_2(0, 1)$ is not spanned by any finite set of functions, nor is the subspace $\mathbb{P}$ of polynomials on $(0, 1)$ of arbitrary degree. In general, there are many different bases for a given linear space. For example, there are numerous useful bases for $\mathbb{P}_{n-1}$, including the monomials, the Lagrange polynomials (sometimes called fundamental polynomials), Legendre polynomials, etc. But it can be shown that every basis for a given linear space contains the same number of vectors. Thus, we can unambiguously define the *dimension* of a linear space to be the number of vectors contained in any basis for it. For example, the space $\mathbb{P}_{n-1}$ has dimension n, whereas $L_2(0, 1)$ is infinite dimensional. For computational purposes, we will be concerned primarily with finite-dimensional subspaces of infinite-dimensional spaces such as L_2.

2.3 Linear Operators

A *linear transformation* from linear space $\mathcal{U}$ to linear space $\mathcal{V}$ is a mapping $T\colon \mathcal{U} \to \mathcal{V}$ such that for any $s, t \in \mathcal{U}$ and $\alpha, \beta \in \mathbb{R}$,

$$T(\alpha s + \beta t) = \alpha T(s) + \beta T(t). \tag{2.6}$$

For example, if $\boldsymbol{V} = [\, v_1 \quad v_2 \quad \cdots \quad v_n \,]$ is a set of vectors in a linear space $\mathcal{V}$, then the mapping $T\colon \mathbb{R}^n \to \mathcal{V}$ defined for $\boldsymbol{x} \in \mathbb{R}^n$ by $T(\boldsymbol{x}) = \boldsymbol{V}\boldsymbol{x}$ is a linear transformation, which by slight abuse of notation we also denote by $\boldsymbol{V}$. When $\mathcal{V} = \mathcal{U}$, a linear transformation is often called a *linear operator* on $\mathcal{U}$, especially when $\mathcal{U}$ is a function space, such as L_2. For example, differentiation and indefinite integration are linear operators on appropriate subspaces of L_2. Another important special case is when $\mathcal{V} = \mathbb{R}$, in which case a linear transformation is called a *linear functional*. For example, if $\mathcal{U}$ is an inner product space and $v \in \mathcal{U}$, then the mapping $f\colon \mathcal{U} \to \mathbb{R}$ defined for $u \in \mathcal{U}$ by $f(u) = \langle u, v \rangle$ is a linear functional.

The set $\mathcal{L}(\mathcal{U}, \mathcal{V})$ of all linear transformations from linear space $\mathcal{U}$ to linear space $\mathcal{V}$ forms a linear space under pointwise addition and scalar multiplication. If $\mathcal{U}$ and $\mathcal{V}$ are normed linear spaces, then a linear transformation $T\colon \mathcal{U} \to \mathcal{V}$ is said to be *bounded* if $\sup_{\|s\|=1} \|T(s)\| < \infty$. The set $\mathcal{B}(\mathcal{U}, \mathcal{V})$ of all such bounded linear transformations forms a normed linear space itself, with norm defined for $T \in \mathcal{B}(\mathcal{U}, \mathcal{V})$ by $\|T\| = \sup_{\|s\|=1} \|T(s)\|$. Further, $\mathcal{B}(\mathcal{U}, \mathcal{V})$ is complete if $\mathcal{V}$ is. In particular, $\mathcal{B}(\mathcal{U}, \mathbb{R})$, the Banach space of all bounded linear functionals on $\mathcal{U}$, is called the *dual* or *conjugate* space of $\mathcal{U}$, denoted by $\mathcal{U}^*$. If $\mathcal{U}$ is a Hilbert space and $f \in \mathcal{U}^*$, then by the Riesz Representation Theorem there is a unique $v \in \mathcal{U}$ such that $f(u) = \langle u, v \rangle$ for any $u \in \mathcal{U}$, and hence we can identify $\mathcal{U}^*$ with $\mathcal{U}$. $L_2(\Omega)$ is self-dual, for example, as every linear functional in $L_2^*(\Omega)$ corresponds to integration against a unique function in $L_2(\Omega)$.

For $T \in \mathcal{B}(\mathcal{U}, \mathcal{V})$, its *adjoint* $T^*\colon \mathcal{V}^* \to \mathcal{U}^*$ is defined for $f \in \mathcal{V}^*$ and $u \in \mathcal{U}$ by

$$[T^*(f)](u) = f(T(u)), \tag{2.7}$$

i.e., if f is a linear functional on $\mathcal{V}$, then $T^*(f)$ is a linear functional on $\mathcal{U}$ whose value for $u \in \mathcal{U}$ is given by $f(T(u))$. It is fairly easy to show that $T^* \in \mathcal{B}(\mathcal{V}^*, \mathcal{U}^*)$ and $\|T^*\| = \|T\|$. If $\mathcal{U}$ and $\mathcal{V}$ are finite dimensional, so that T is represented by a matrix, then T^* is represented by its transpose (or conjugate transpose in the complex case). The adjoint obeys the familiar properties of transposition, such as $(ST)^* = T^* S^*$, where $T \in \mathcal{B}(\mathcal{U}, \mathcal{V})$ and $S \in \mathcal{B}(\mathcal{V}, \mathcal{W})$.

If $\mathcal{U}$ and $\mathcal{V}$ are Hilbert spaces, and hence identified with $\mathcal{U}^*$ and $\mathcal{V}^*$, respectively, then from (2.7) we see that the adjoint $T^*\colon \mathcal{V} \to \mathcal{U}$ is characterized by the relationship

$$\langle u, T^*(v) \rangle = \langle T(u), v \rangle, \tag{2.8}$$

which must hold for all $u \in \mathcal{U}$, $v \in \mathcal{V}$. For example, if $\boldsymbol{V} = [\, v_1 \quad v_2 \quad \cdots \quad v_n \,]$ is a set of functions in $L_2(\Omega)$, then as we have seen, a linear transformation $\boldsymbol{V}\colon \mathbb{R}^n \to L_2(\Omega)$ is defined for $\boldsymbol{x} \in \mathbb{R}^n$ by $\boldsymbol{V}\boldsymbol{x} = \sum_{i=1}^n x_i v_i$. For any $\boldsymbol{x} \in \mathbb{R}^n$ and $f \in L_2(\Omega)$, we have

$$\begin{aligned} \langle \boldsymbol{V}\boldsymbol{x}, f \rangle &= \int_\Omega (\boldsymbol{V}\boldsymbol{x})(f) = \int_\Omega \Big(\sum_{i=1}^n x_i v_i \Big) f \\ &= \sum_{i=1}^n x_i \int_\Omega v_i f = \sum_{i=1}^n \langle v_i, f \rangle x_i = \langle \boldsymbol{x}, \boldsymbol{V}^* f \rangle, \end{aligned} \tag{2.9}$$

and hence for $f \in L_2(\Omega)$ the value of the adjoint $\boldsymbol{V}^*\colon L_2(\Omega) \to \mathbb{R}^n$ is given by

$$\boldsymbol{V}^* f = \begin{bmatrix} \langle v_1, f \rangle \\ \vdots \\ \langle v_n, f \rangle \end{bmatrix} \in \mathbb{R}^n. \tag{2.10}$$

More generally, if $\boldsymbol{F} = [\, f_1 \quad f_2 \quad \cdots \quad f_k \,]$ is a set of functions in $L_2(\Omega)$, then

$$\boldsymbol{V}^* \boldsymbol{F} = \begin{bmatrix} \langle v_1, f_1 \rangle & \cdots & \langle v_1, f_k \rangle \\ \vdots & \ddots & \vdots \\ \langle v_n, f_1 \rangle & \cdots & \langle v_n, f_k \rangle \end{bmatrix} \in \mathbb{R}^{n \times k}. \tag{2.11}$$

In particular, the *Gram matrix*

$$\boldsymbol{V}^* \boldsymbol{V} = \begin{bmatrix} \langle v_1, v_1 \rangle & \cdots & \langle v_1, v_n \rangle \\ \vdots & \ddots & \vdots \\ \langle v_n, v_1 \rangle & \cdots & \langle v_n, v_n \rangle \end{bmatrix} \in \mathbb{R}^{n \times n} \tag{2.12}$$

is symmetric, and furthermore it is nonsingular (in fact, positive definite) if, and only if, the set of functions $\boldsymbol{V}$ is linearly independent. The importance of the Gram matrix will soon become apparent.

2.4 Orthogonality and Projectors

Two vectors u_1 and u_2 in an inner product space $\mathcal{U}$ are said to be *orthogonal* if $\langle u_1, u_2 \rangle = 0$. A (not necessarily finite) set of vectors $\{u_i\}$ is said to be *orthonormal* if

$$\begin{aligned} \langle u_i, u_j \rangle &= 0, \quad i \neq j, \\ \langle u_i, u_j \rangle &= 1, \quad i = j, \end{aligned}$$

i.e., the vectors are pairwise orthogonal and each has norm 1. In particular, a finite set of functions $\boldsymbol{V} = [\, v_1 \quad v_2 \quad \cdots \quad v_n \,] \subseteq L_2(\Omega)$ is orthonormal if $\boldsymbol{V}^*\boldsymbol{V} = \boldsymbol{I}$, the $n \times n$ identity matrix. A set of orthonormal functions is necessarily independent, since $\boldsymbol{V}\boldsymbol{x} = \boldsymbol{0} \Rightarrow \boldsymbol{V}^*\boldsymbol{V}\boldsymbol{x} = \boldsymbol{I}\boldsymbol{x} = \boldsymbol{x} = \boldsymbol{0}$. For any set $S \subseteq \mathcal{U}$, its

orthogonal complement

$$S^\perp = \{u \in \mathcal{U} : \langle u, v\rangle = 0 \quad \forall\, v \in S\} \tag{2.13}$$

is the subspace composed of all vectors in $\mathcal{U}$ that are orthogonal to every vector in S.

A linear operator $P \in \mathcal{B}(\mathcal{U},\mathcal{U})$ is said to be a *projector* if it is idempotent, i.e., $P^2 = P$. Such a projector takes any vector $u \in \mathcal{U}$ into a subspace of $\mathcal{U}$, but leaves any vector already in that subspace unchanged. For example, polynomial interpolation at a given set of points $0 \le t_1 < t_2 < \cdots < t_n \le 1$ is a projector on $L_2(0,1)$ that maps any function $f \in L_2(0,1)$ to the polynomial $P(f) \in \mathbb{P}_{n-1} \subseteq L_2(0,1)$ given by

$$[P(f)](t) = \sum_{i=1}^{n} \left(\prod_{j \ne i} \frac{t - t_j}{t_i - t_j} \right) f(t_i). \tag{2.14}$$

A projector P that is also self-adjoint, i.e., $P^* = P$, is said to be an *orthogonal projector*, in the sense that for any $u \in \mathcal{U}$,

$$\langle u - P(u), P(u)\rangle = \langle P(u - P(u)), u\rangle = \langle P(u) - P^2(u), u\rangle = \langle P(u) - P(u), u\rangle = 0. \tag{2.15}$$

One of the most important uses of orthogonal projectors is to compute the best approximation to a given vector by a vector from some subspace. For example, consider the subspace of $L_2(\Omega)$ spanned by a linearly independent set of functions $\boldsymbol{V} = [\, v_1 \quad v_2 \quad \cdots \quad v_n \,]$. As we have seen, a linear transformation $\boldsymbol{V}\colon \mathbb{R}^n \to L_2(\Omega)$ is defined for $\boldsymbol{x} \in \mathbb{R}^n$ by $\boldsymbol{V}\boldsymbol{x} = \sum_{i=1}^n x_i v_i$, and the value of the adjoint $\boldsymbol{V}^*\colon L_2(\Omega) \to \mathbb{R}^n$ is given for $f \in L_2(\Omega)$ by

$$\boldsymbol{V}^* f = \begin{bmatrix} \langle v_1, f\rangle \\ \vdots \\ \langle v_n, f\rangle \end{bmatrix}. \tag{2.16}$$

Owing to the linear independence of the functions v_i, the $n \times n$ Gram matrix $\boldsymbol{V}^*\boldsymbol{V}$ is nonsingular, and thus the operator $P\colon L_2(\Omega) \to L_2(\Omega)$ given for $f \in L_2(\Omega)$ by

$$P(f) = \boldsymbol{V}(\boldsymbol{V}^*\boldsymbol{V})^{-1}\boldsymbol{V}^* f \tag{2.17}$$

is well defined. It is easily seen that $P^2 = P = P^*$, i.e., P is idempotent and self-adjoint, so P is an orthogonal projector onto $\operatorname{span}(\boldsymbol{V})$. To confirm that $P(f)$ indeed gives the best approximation to f from $\operatorname{span}(\boldsymbol{V})$ with respect to the norm of the induced inner product, we minimize the function

$$\phi(\boldsymbol{x}) = \tfrac{1}{2}\|f - \boldsymbol{V}\boldsymbol{x}\|^2 = \tfrac{1}{2}\int_\Omega \Big(f - \sum_{j=1}^n x_j v_j\Big)^2 \tag{2.18}$$

by setting each component of its gradient to zero, i.e., for $i = 1, \ldots, n$,

$$0 = \frac{\partial \phi}{\partial x_i} = \int_\Omega \Big(f - \sum_{j=1}^n x_j v_j \Big)(-v_i) = -\int_\Omega v_i f + \sum_{j=1}^n x_j \int_\Omega v_i v_j$$
$$= -\langle v_i, f \rangle + \sum_{j=1}^n x_j \langle v_i, v_j \rangle, \tag{2.19}$$

so that $\boldsymbol{x}$ is the solution to the equation $\boldsymbol{V}^* \boldsymbol{V} \boldsymbol{x} = \boldsymbol{V}^* f$, or $\boldsymbol{x} = (\boldsymbol{V}^* \boldsymbol{V})^{-1} \boldsymbol{V}^* f$, and hence $P(f) = \boldsymbol{V}(\boldsymbol{V}^* \boldsymbol{V})^{-1} \boldsymbol{V}^* f = \boldsymbol{V} \boldsymbol{x}$ is the best approximation, as claimed. Note that if the functions in $\boldsymbol{V}$ were orthonormal, then we would have $\boldsymbol{V}^* \boldsymbol{V} = \boldsymbol{I}$, and the orthogonal projector would simply be given by $P = \boldsymbol{V} \boldsymbol{V}^*$.

Now suppose that $\boldsymbol{V}$ is a subset of a subspace of dimension $m > n$ spanned by a set of linearly independent functions $\boldsymbol{U} = [\, u_1 \quad u_2 \quad \cdots \quad u_m \,] \subseteq L_2(\Omega)$. Then each function in $\boldsymbol{V}$ can be expressed as a linear combination $v_j = \sum_{i=1}^m w_{ij} u_i$, $j = 1, \ldots, n$, or in matrix notation $\boldsymbol{V} = \boldsymbol{U} \boldsymbol{W}$, where $\boldsymbol{W}$ is an $m \times n$ matrix with entries w_{ij}. Expressed in terms of the functions in $\boldsymbol{U}$, the orthogonal projector onto span$(\boldsymbol{V})$ becomes

$$P = \boldsymbol{U} \boldsymbol{W} (\boldsymbol{W}^* \boldsymbol{U}^* \boldsymbol{U} \boldsymbol{W})^{-1} \boldsymbol{W}^* \boldsymbol{U}^*. \tag{2.20}$$

3 Linear Systems

Discretization of linear differential equations with nontrivial null spaces, as we will discuss in greater detail below, leads to rank deficient systems of linear algebraic equations, so we begin with a general discussion of the latter.

3.1 Compatibility and Deficiency

The definitions and explanations at the beginning of this section are well-known in linear algebra contexts and are discussed in comprehensive texts such as [29, 35]. We include them here for completeness.

Consider a general system of linear algebraic equations

$$\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}, \tag{3.1}$$

where $\boldsymbol{A} \in \mathbb{R}^{m\times p}$ and $\boldsymbol{b} \in \mathbb{R}^m$ are known, and $\boldsymbol{x} \in \mathbb{R}^p$ is to be determined. We allow arbitrary values for m and p, so the system (3.1) may be overdetermined ($m > p$), underdetermined ($m < p$), or square ($m = p$). For now, we assume that $k = \text{rank}(\boldsymbol{A})$ is known; rank determination is often problematic in practice, but it can be clear cut in some applications, based on structural considerations, for example. We necessarily have $k \leq \min\{m, p\}$, but we allow the possibility that $k < \min\{m, p\}$, i.e., $\boldsymbol{A}$ may not have full row rank or full column rank.

If $\boldsymbol{b} \in \text{span}(\boldsymbol{A})$, then the system (3.1) is said to be *compatible*, in which case at least one solution exists. If $\boldsymbol{A}$ has full row rank ($k = m$), then $\text{span}(\boldsymbol{A}) = \mathbb{R}^m$, so that compatibility is guaranteed for any $\boldsymbol{b}$. If $k < m$, on the other hand, then compatibility depends on the particular value for $\boldsymbol{b}$.

If the system (3.1) is incompatible, i.e., $\boldsymbol{b} \notin \text{span}(\boldsymbol{A})$, then there is no solution in the conventional sense, but the notion of solution can be usefully extended by treating (3.1) as a least squares problem, i.e., we seek a vector $\boldsymbol{x}$ that minimizes the residual $\|\boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}\|_2$. Such a least squares solution *always* exists, since the function $\phi(\boldsymbol{y}) = \|\boldsymbol{b} - \boldsymbol{y}\|_2$ is continuous and coercive on $\mathbb{R}^m$, and hence has a minimum on the closed, unbounded set $\text{span}(\boldsymbol{A})$, i.e., there is a vector $\boldsymbol{y} \in \text{span}(\boldsymbol{A})$ closest to $\boldsymbol{b}$ in the 2-norm. For compatible systems, least squares and conventional solutions coincide.

If $\boldsymbol{A}$ has a nontrivial null space, then the system (3.1) is said to be *deficient*, in which case no solution can be unique, since for any (conventional or least squares) solution $\boldsymbol{x}$, $\boldsymbol{x} + \boldsymbol{z}$ is also a solution (in the same sense), where $\boldsymbol{z}$ is

any nonzero null vector. If $\boldsymbol{A}$ does not have full column rank ($k < p$), then the system (3.1) is necessarily deficient, and this is always the case if (3.1) is underdetermined ($m < p$). If $m \geq p$, on the other hand, then (3.1) may or may not be deficient.

We will analyze the compatibility and deficiency of (3.1) in greater detail using the singular value decomposition (SVD)

$$\boldsymbol{A} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T, \tag{3.2}$$

where $\boldsymbol{U}$ is an $m \times m$ orthogonal matrix, $\boldsymbol{V}$ is an $p \times p$ orthogonal matrix, and $\boldsymbol{\Sigma}$ is an $m \times p$ diagonal matrix with nonnegative diagonal entries σ_i, ordered so that $\sigma_{i-1} \geq \sigma_i$, $i = 2, \ldots, \min\{m, p\}$. Since $\boldsymbol{A}$ has rank k, we can write (3.2) in the partitioned form

$$\boldsymbol{A} = [\,\boldsymbol{U}_1 \quad \boldsymbol{U}_2\,] \begin{bmatrix} \boldsymbol{\Sigma}_1 & \boldsymbol{O} \\ \boldsymbol{O} & \boldsymbol{O} \end{bmatrix} [\,\boldsymbol{V}_1 \quad \boldsymbol{V}_2\,]^T = \boldsymbol{U}_1\boldsymbol{\Sigma}_1\boldsymbol{V}_1^T, \tag{3.3}$$

where $\boldsymbol{U}_1$ is $m \times k$, $\boldsymbol{\Sigma}_1$ is $k \times k$ and nonsingular, and $\boldsymbol{V}_1$ is $p \times k$. The columns of $\boldsymbol{U}_1$ form an orthonormal basis for $\text{span}(\boldsymbol{A}) = \{\boldsymbol{A}\boldsymbol{x} : \boldsymbol{x} \in \mathbb{R}^p\}$, and the columns of $\boldsymbol{U}_2$ form an orthonormal basis for its orthogonal complement, $\text{span}(\boldsymbol{A})^\perp$. Similarly, the columns of $\boldsymbol{V}_2$ form an orthonormal basis for the null space of $\boldsymbol{A}$, $N(\boldsymbol{A}) = \{\boldsymbol{z} \in \mathbb{R}^p : \boldsymbol{A}\boldsymbol{z} = \boldsymbol{0}\}$, and the columns of $\boldsymbol{V}_1$ form an orthonormal basis for its orthogonal complement, $N(\boldsymbol{A})^\perp$.

To help in analyzing (3.1), we define the vector $\boldsymbol{c} \in \mathbb{R}^m$ by

$$\boldsymbol{c} = \boldsymbol{U}^T\boldsymbol{b} = [\,\boldsymbol{U}_1 \quad \boldsymbol{U}_2\,]^T\,\boldsymbol{b} = \begin{bmatrix} \boldsymbol{U}_1^T \\ \boldsymbol{U}_2^T \end{bmatrix} \boldsymbol{b} = \begin{bmatrix} \boldsymbol{U}_1^T\boldsymbol{b} \\ \boldsymbol{U}_2^T\boldsymbol{b} \end{bmatrix} = \begin{bmatrix} \boldsymbol{c}_1 \\ \boldsymbol{c}_2 \end{bmatrix} \tag{3.4}$$

and the vector $\boldsymbol{y} \in \mathbb{R}^p$ by

$$\boldsymbol{y} = \boldsymbol{V}^T\boldsymbol{x} = [\,\boldsymbol{V}_1 \quad \boldsymbol{V}_2\,]^T\,\boldsymbol{x} = \begin{bmatrix} \boldsymbol{V}_1^T \\ \boldsymbol{V}_2^T \end{bmatrix} \boldsymbol{x} = \begin{bmatrix} \boldsymbol{V}_1^T\boldsymbol{x} \\ \boldsymbol{V}_2^T\boldsymbol{x} \end{bmatrix} = \begin{bmatrix} \boldsymbol{y}_1 \\ \boldsymbol{y}_2 \end{bmatrix}, \tag{3.5}$$

where $\boldsymbol{c}_1, \boldsymbol{y}_1 \in \mathbb{R}^k$. Using (3.2)-(3.5), the system (3.1) becomes

$$\begin{aligned}
\boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T\boldsymbol{x} &= \boldsymbol{b}, && (3.6)\\
\boldsymbol{\Sigma}\boldsymbol{V}^T\boldsymbol{x} &= \boldsymbol{U}^T\boldsymbol{b}, && (3.7)\\
\boldsymbol{\Sigma}\boldsymbol{y} &= \boldsymbol{c}, && (3.8)\\
& && (3.9)\\
\begin{bmatrix} \boldsymbol{\Sigma}_1 & \boldsymbol{O} \\ \boldsymbol{O} & \boldsymbol{O} \end{bmatrix} \begin{bmatrix} \boldsymbol{y}_1 \\ \boldsymbol{y}_2 \end{bmatrix} &= \begin{bmatrix} \boldsymbol{c}_1 \\ \boldsymbol{c}_2 \end{bmatrix}, && (3.10)\\
& && (3.11)\\
\begin{bmatrix} \boldsymbol{\Sigma}_1\boldsymbol{y}_1 \\ \boldsymbol{0} \end{bmatrix} &= \begin{bmatrix} \boldsymbol{c}_1 \\ \boldsymbol{c}_2 \end{bmatrix}. && (3.12)
\end{aligned}$$

From (3.12), we see that (3.1) is compatible, so that a solution exists in the

conventional sense, only if

$$\boldsymbol{c}_2 = \boldsymbol{U}_2^T \boldsymbol{b} = \boldsymbol{0}, \tag{3.13}$$

which means that the right-hand side $\boldsymbol{b}$ must be orthogonal to $\text{span}(\boldsymbol{A})^\perp = N(\boldsymbol{A}^T)$. This *compatibility condition* comes as no surprise: by definition, the system (3.1) has a solution in the conventional sense if, and only if, $\boldsymbol{b} \in \text{span}(\boldsymbol{A})$, and hence $\boldsymbol{b}$ cannot have a nonzero component in $\text{span}(\boldsymbol{A})^\perp$ if such a solution is to exist.

In any case, a solution (conventional if compatible, least squares otherwise) to (3.1) is given by

$$\boldsymbol{x} = \boldsymbol{V}\boldsymbol{y} = \begin{bmatrix} \boldsymbol{V}_1 & \boldsymbol{V}_2 \end{bmatrix} \begin{bmatrix} \boldsymbol{y}_1 \\ \boldsymbol{y}_2 \end{bmatrix} = \boldsymbol{V}_1 \boldsymbol{y}_1 + \boldsymbol{V}_2 \boldsymbol{y}_2, \tag{3.14}$$

where $\boldsymbol{y}_1 = \boldsymbol{\Sigma}_1^{-1} \boldsymbol{c}_1$ and $\boldsymbol{y}_2$ can be chosen arbitrarily. Again, this result agrees with expectations, as the number of free parameters, namely the $p - k$ components of $\boldsymbol{y}_2$, reflects the dimension of $N(\boldsymbol{A})$, and a solution cannot be unique unless $k = p$.

From (3.14) we see that we can single out any particular solution of a deficient system by choosing $\boldsymbol{y}_2$ appropriately. For example, a null-space component of zero ($\boldsymbol{y}_2 = \boldsymbol{0}$) yields the (unique) solution $\boldsymbol{x}$ having minimum 2-norm. More generally, if the desired null-space component of the solution is $\boldsymbol{d}$, i.e., $\boldsymbol{V}_2 \boldsymbol{y}_2 = \boldsymbol{d}$, then we choose $\boldsymbol{y}_2 = \boldsymbol{V}_2^T \boldsymbol{d}$. Combining the latter prescription with the determination of $\boldsymbol{y}_1$, we obtain the $(p + k) \times p$ system

$$\begin{bmatrix} \boldsymbol{\Sigma}_1 & \boldsymbol{O} \\ \boldsymbol{O} & \boldsymbol{V}_2 \end{bmatrix} \begin{bmatrix} \boldsymbol{y}_1 \\ \boldsymbol{y}_2 \end{bmatrix} = \begin{bmatrix} \boldsymbol{c}_1 \\ \boldsymbol{d} \end{bmatrix} \tag{3.15}$$

for $\boldsymbol{y}$, which in turn determines $\boldsymbol{x} = \boldsymbol{V}\boldsymbol{y}$ via (3.5). The system (3.15) can be related to the original system (3.1) by using (3.4) and (3.5) to obtain the equivalent $(p + k) \times p$ system

$$\begin{bmatrix} \boldsymbol{\Sigma}_1 & \boldsymbol{O} \\ \boldsymbol{O} & \boldsymbol{V}_2 \end{bmatrix} \boldsymbol{V}\boldsymbol{x} = \begin{bmatrix} \boldsymbol{\Sigma}_1 \boldsymbol{V}_1^T \\ \boldsymbol{V}_2 \boldsymbol{V}_2^T \end{bmatrix} \boldsymbol{x} = \begin{bmatrix} \boldsymbol{U}_1^T \boldsymbol{b} \\ \boldsymbol{d} \end{bmatrix}, \tag{3.16}$$

and then premultiplying the upper block row of (3.16) by $\boldsymbol{U}_1$ to arrive at the $(m + p) \times p$ system

$$\begin{bmatrix} \boldsymbol{U}_1 \boldsymbol{\Sigma}_1 \boldsymbol{V}_1^T \\ \boldsymbol{V}_2 \boldsymbol{V}_2^T \end{bmatrix} \boldsymbol{x} = \begin{bmatrix} \boldsymbol{A} \\ \boldsymbol{V}_2 \boldsymbol{V}_2^T \end{bmatrix} \boldsymbol{x} = \begin{bmatrix} \boldsymbol{b} \\ \boldsymbol{d} \end{bmatrix}. \tag{3.17}$$

The matrix $\boldsymbol{V}_2 \boldsymbol{V}_2^T$ is symmetric and idempotent, and hence it is an orthogonal projector onto $N(\boldsymbol{A})$. Thus, the upper block row of (3.17) ensures that the solution $\boldsymbol{x}$ satisfies the original system (3.1), and the lower block row of (3.17) ensures that $\boldsymbol{x}$ has the desired null-space component. Premultiplying the lower

row of (3.17) by $\boldsymbol{V}_2^T$ yields the smaller $(m+p-k)\times p$ system

$$\begin{bmatrix} \boldsymbol{A} \\ \boldsymbol{V}_2^T \end{bmatrix} \boldsymbol{x} = \begin{bmatrix} \boldsymbol{b} \\ \boldsymbol{V}_2^T \boldsymbol{d} \end{bmatrix}, \tag{3.18}$$

which is easily shown to have full column rank, and hence it determines the desired solution $\boldsymbol{x}$ uniquely. Moreover, if the original system (3.1) is compatible, then the system (3.18) is also compatible.

An extremely simple yet instructive example is the $m \times p$ system

$$\begin{bmatrix} \boldsymbol{I}_k & \boldsymbol{O} \\ \boldsymbol{O} & \boldsymbol{O} \end{bmatrix} \boldsymbol{x} = \begin{bmatrix} \boldsymbol{b}_1 \\ \boldsymbol{b}_2 \end{bmatrix} = \boldsymbol{b},$$

where $\boldsymbol{I}_k$ denotes the identity matrix of dimension k. For this example, we have

$$\boldsymbol{U}_2 = \begin{bmatrix} \boldsymbol{O} \\ \boldsymbol{I}_{m-k} \end{bmatrix} \qquad \text{and} \qquad \boldsymbol{V}_2 = \begin{bmatrix} \boldsymbol{O} \\ \boldsymbol{I}_{p-k} \end{bmatrix}.$$

Thus, the compatibility condition is

$$\boldsymbol{c}_2 = \boldsymbol{U}_2^T \boldsymbol{b} = [\, \boldsymbol{O} \quad \boldsymbol{I}_{m-k} \,] \begin{bmatrix} \boldsymbol{b}_1 \\ \boldsymbol{b}_2 \end{bmatrix} = \boldsymbol{b}_2 = \boldsymbol{0},$$

and the orthogonal projector onto $N(\boldsymbol{A})$ is

$$\boldsymbol{V}_2 \boldsymbol{V}_2^T = \begin{bmatrix} \boldsymbol{O} & \boldsymbol{O} \\ \boldsymbol{O} & \boldsymbol{I}_{p-k} \end{bmatrix}.$$

Therefore, for any compatible instance of this example and any desired choice

$$\boldsymbol{d} = \begin{bmatrix} \boldsymbol{0} \\ \boldsymbol{d}_2 \end{bmatrix}$$

for the null-space component of the solution, the system (3.17) becomes

$$\begin{bmatrix} \boldsymbol{A} \\ \boldsymbol{V}_2 \boldsymbol{V}_2^T \end{bmatrix} \boldsymbol{x} = \begin{bmatrix} \boldsymbol{I}_k & \boldsymbol{O} \\ \boldsymbol{O} & \boldsymbol{O} \\ \boldsymbol{O} & \boldsymbol{O} \\ \boldsymbol{O} & \boldsymbol{I}_{n-k} \end{bmatrix} \boldsymbol{x} = \begin{bmatrix} \boldsymbol{b}_1 \\ \boldsymbol{0} \\ \boldsymbol{0} \\ \boldsymbol{d}_2 \end{bmatrix} = \begin{bmatrix} \boldsymbol{b} \\ \boldsymbol{d} \end{bmatrix},$$

and the smaller system (3.18) becomes

$$\begin{bmatrix} \boldsymbol{A} \\ \boldsymbol{V}_2^T \end{bmatrix} \boldsymbol{x} = \begin{bmatrix} \boldsymbol{I}_k & \boldsymbol{O} \\ \boldsymbol{O} & \boldsymbol{O} \\ \boldsymbol{O} & \boldsymbol{I}_{n-k} \end{bmatrix} \boldsymbol{x} = \begin{bmatrix} \boldsymbol{b}_1 \\ \boldsymbol{0} \\ \boldsymbol{d}_2 \end{bmatrix} = \begin{bmatrix} \boldsymbol{b} \\ \boldsymbol{V}_2^T \boldsymbol{d} \end{bmatrix},$$

and in either case the solution is given by

$$\boldsymbol{x} = \begin{bmatrix} \boldsymbol{b}_1 \\ \boldsymbol{d}_2 \end{bmatrix}.$$

3.2 Solutions of Deficient Systems

Motivated by discretizations of differential operators, we will be concerned primarily with systems that are deficient (but usually compatible), and we will exploit the resulting nonuniqueness of solutions to determine a specific solution satisfying additional desired conditions. We next list some useful criteria for singling out a particular solution for an $m \times p$ system $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$ of rank $k < p$ and discuss the mathematical or physical rationale for making a particular choice. The various discretization methodologies alluded to will be discussed in greater detail in Chapter 5.

Specified null-space component. We saw in Section 3.1 that specifying the null-space component of the solution provides a completely general way to single out any particular solution for a deficient linear system, and it is a very natural way to do so in certain contexts. To give a simple example, for a differential operator that annihilates all constant functions, we might require the discretized solution to have a desired mean, which fixes that one degree of freedom (there will still be others, of course, if $p - k > 1$). This approach requires an explicit basis for $N(\boldsymbol{A})$, which can be computed using the SVD as in Section 3.1, or somewhat less expensively using QR factorization with column pivoting, but these approaches may be computationally infeasible for large sparse problems. Fortunately, alternative methods are available that exploit and preserve sparsity—to the extent possible—in computing a null basis. Although the resulting basis may not be orthonormal, it can still be used to specify the null-space component of the solution to the linear system, as we will demonstrate later.

Partially specified solution. Another way to remove degrees of freedom from a deficient system is to specify a selected portion of the solution values directly, which is done in conventional practice by imposing boundary conditions. While relevant boundary conditions arise naturally in many contexts, it is not always obvious how to prescribe consistent boundary conditions to achieve a desired objective, nor is it always desirable to do so. Our main goal is to explore alternatives to the usual specification of boundary conditions, so we will not dwell on this approach, but it is important to understand that this is merely one of several viable options, and it is not always the best in all situations.

Minimum-norm solution. In many contexts, such as data fitting, a commonly used criterion for selecting a specific solution for a deficient system is to choose the solution having minimum norm. For both theoretical and practical reasons, the norm chosen is usually the 2-norm, in which case the minimum-norm solution is always unique. This criterion is of interest on physical grounds, as minimizing the norm of the solution often corresponds to minimizing the en-

ergy or power of the solution in appropriate contexts. Several methods are available for computing the minimum-norm solution, including the pseudoinverse, the SVD, and QR factorization, but again none of these is attractive for large sparse systems. As we saw in Section 3.1, however, specifying a null-space component of zero yields the minimum-norm solution, so this criterion is a special case of specifying the null-space component and can use the same type of sparse null basis already cited there.

Basic solution. Another approach that is often used, for example in data-fitting problems, is to choose a *basic* solution, which by definition has at most k nonzero components. The motivation for this criterion is to retain only those components of the solution that are well determined by the data, and to suppress those that are not. In polynomial data-fitting, for example, the low-degree terms may be well determined, but higher-degree terms may not be well resolved by the data and hence are dropped. A common way of computing a basic solution in this context is to use QR factorization with column pivoting to identify the most independent components. In our context, this approach could be used, for example, to filter out specific modes or frequencies in a spectral discretization. Observe, however, that a basic solution is still not unique, in general, since there are many choices for which k components should be nonzero, so additional constraints may still be needed.

Other constraints. A variety of other constraints can be useful in certain contexts. For example, instead of specifying solution components directly, as already discussed, one could specify relationships among solution components, analogous to imposing Neumann or Robin boundary conditions instead of Dirichlet boundary conditions. In physical problems where conservation of mass or energy is important, one might want a solution having prescribed norm. For the 2-norm, this is a quadratic rather than linear constraint, however, which significantly complicates the computation. More specialized constraints may be relevant for particular differential operators. For example, requiring the solution to be solenoidal (i.e., divergence-free) removes ambiguity in seeking a function having given curl.

Combinations of criteria. Some of the criteria we have discussed reduce but do not entirely eliminate the freedom in determining a solution, leaving open the possibility of using multiple criteria in combination to specify a solution uniquely.

4 Null-Space Approach

In this thesis we present methods for solving differential equations using a specified null-space component to single out a particular solution, as described in Section 3.2, and develop various aspects of the solution process necessary to make this approach viable computationally. This approach to posing and solving differential equations was suggested for a specific application by Acharya in [2] and, to the best of our knowledge, has not been discussed elsewhere in the literature. In this chapter we describe the general null-space approach in the continuous realm, which is the basis for all our discrete methods. We provide examples with one-dimensional functions and discuss how the null-space approach relates to conventional methods.

4.1 Overview of Problem Formulation

Our goal is to solve linear differential equations in cases where the differential operator has a nontrivial null space. Let Ω be the domain on which the problem is defined, and let $\mathcal{D}$ be our differential operator. Given $\alpha \in C_0(\Omega)$ (continuous functions over our domain), we wish to solve the equation

$$\mathcal{D}u = \alpha, \tag{4.1}$$

i.e., we wish to find a function $u \in C_k(\Omega)$ such that the differential operator $\mathcal{D}$ maps u to α, where $k > 0$ is the order of $\mathcal{D}$.

By assumption, $\mathcal{D}$ has a nontrivial null space, $N(\mathcal{D})$, and hence the solution to (4.1) cannot be unique. We will select one solution u from the infinite family of solutions by requiring the null space component of u to be either (1) the projection of some given function β onto $N(\mathcal{D})$ or (2) an explicit null space component specified by a given vector γ. For (1), we require that $\mathcal{P}u = \mathcal{P}\beta$, where $\mathcal{P}$ is the orthogonal projector onto $N(\mathcal{D})$ in the usual inner product on $L_2(\Omega)$ as described in Chapter 2. Thus, we wish to solve the *augmented* problem

$$\begin{bmatrix} \mathcal{D} \\ \mathcal{P} \end{bmatrix} u = \begin{bmatrix} \alpha \\ \mathcal{P}\beta \end{bmatrix}, \tag{4.2}$$

whose solution u is now unique. For (2), we require that $\mathcal{P}u = \boldsymbol{V}\gamma$, where $\mathcal{P}$ is the orthogonal projector onto $N(\mathcal{D})$ as above and $\boldsymbol{V}$ contains the set of basis functions for $N(\mathcal{D})$ (or a finite-dimensional approximation to it if $N(\mathcal{D})$

is infinite-dimensional) used to form $\mathcal{P}$. Here, we wish to solve the *augmented* problem

$$\begin{bmatrix} \mathcal{D} \\ \mathcal{P} \end{bmatrix} u = \begin{bmatrix} \alpha \\ \boldsymbol{V}\gamma \end{bmatrix}, \tag{4.3}$$

whose solution u is now unique. Each particular application will dictate the form of the null-space constraint. In most of our examples, we use a constraint of the first kind and solve Equation (4.4), which is valid as explained below.

We need not use the full projector $\mathcal{P} = \boldsymbol{V}(\boldsymbol{V}^*\boldsymbol{V})^{-1}\boldsymbol{V}^*$ in either case above, as the solution to

$$\begin{bmatrix} \mathcal{D} \\ \boldsymbol{V}^* \end{bmatrix} u = \begin{bmatrix} \alpha \\ \boldsymbol{V}^*\beta \end{bmatrix} \tag{4.4}$$

necessarily satisfies (4.2), and the solution to

$$\begin{bmatrix} \mathcal{D} \\ \boldsymbol{V}^* \end{bmatrix} u = \begin{bmatrix} \alpha \\ \boldsymbol{V}^*\boldsymbol{V}\gamma \end{bmatrix} \tag{4.5}$$

necessarily satisfies (4.3). In fact, Equations (4.2) and (4.4) are equivalent, as the following lemma shows. Similarly, Equations (4.3) and (4.5) are equivalent.

Lemma 1 *For any matrix $\boldsymbol{V}$ of linearly independent functions and $\mathcal{P} = \boldsymbol{V}(\boldsymbol{V}^*\boldsymbol{V})^{-1}\boldsymbol{V}^*$, $\mathcal{P}u = \mathcal{P}\beta$ if and only if $\boldsymbol{V}^*u = \boldsymbol{V}^*\beta$.*

Proof: Suppose $\boldsymbol{V}^*u = \boldsymbol{V}^*\beta$. Then $\boldsymbol{V}(\boldsymbol{V}^*\boldsymbol{V})^{-1}\boldsymbol{V}^*u = \boldsymbol{V}(\boldsymbol{V}^*\boldsymbol{V})^{-1}\boldsymbol{V}^*\beta$, by applying the same process to both sides of the equation. Thus, $\mathcal{P}u = \mathcal{P}\beta$.

Now suppose $\mathcal{P}u = \mathcal{P}\beta$. First we recall that by definition a linear combination of linearly independent functions equal to zero implies that all the coefficients are zero. Thus, $\boldsymbol{V}\boldsymbol{x} = \boldsymbol{V}\boldsymbol{y}$ implies that $\boldsymbol{V}(\boldsymbol{x}-\boldsymbol{y}) = 0$, which implies $\boldsymbol{x} = \boldsymbol{y}$. Therefore $\boldsymbol{V}(\boldsymbol{V}^*\boldsymbol{V})^{-1}\boldsymbol{V}^*u = \boldsymbol{V}(\boldsymbol{V}^*\boldsymbol{V})^{-1}\boldsymbol{V}^*\beta$ implies $(\boldsymbol{V}^*\boldsymbol{V})^{-1}\boldsymbol{V}^*u = (\boldsymbol{V}^*\boldsymbol{V})^{-1}\boldsymbol{V}^*\beta$. Next, we multiply both sides by the nonsingular matrix $\boldsymbol{V}^*\boldsymbol{V}$ and we find $\boldsymbol{V}^*u = \boldsymbol{V}^*\beta$, as desired. □

In some simple cases, such as the examples of Section 4.2 below, our problem can be solved analytically. In general, we will solve the problem in an approximate sense using one of several discretization techniques described in Chapter 5.

4.2 Examples in 1-D

To illustrate the null space method in the continuous realm, we present an example using the first derivative operator in one dimension. We define our operator, domain, and right-hand-side function α as follows:

$$\begin{aligned} \mathcal{D} &= \frac{d}{dt}, \\ \Omega &= (0,1), \\ \alpha &= 3. \end{aligned}$$

This is not an unusual problem to solve. Written as we would see it in a calculus textbook, this problem is $u'(t) = 3$. The solution is familiar: $u(t) = 3t+c$, where c is any real number. The arbitrary constant c indicates the non-uniqueness of our solution. We need more information in order to single out a unique solution. The usual method for selecting a unique solution would be to specify its value on one of the boundaries of our domain. Using our null-space method, we instead specify a function β to be projected onto the null space, and we force the projection of our solution onto the null space to be the same. That is, we set

$$\boldsymbol{V}^* u = \boldsymbol{V}^* \beta, \tag{4.6}$$

as in (4.4), where $\boldsymbol{V}$ contains functions that form a basis for the null space of the first derivative operator. The null space for this simple problem is the space of constant functions, and a basis for this space is the function $v(t) = 1$. It follows that $\boldsymbol{V} = [1]$ and $\boldsymbol{V}^* f = \int_0^1 1 \cdot f \, \mathrm{dt}$, the mean value of f on $(0, 1)$.

Suppose $\beta = 1$. Equation (4.6) then becomes

$$\int_0^1 1 \cdot (3t + c) \, \mathrm{dt} = \int_0^1 1 \cdot 1 \, \mathrm{dt}.$$

Solving, we find

$$\begin{aligned} \frac{3}{2}t^2 + ct|_0^1 &= t|_0^1, \\ \frac{3}{2} + c &= 1, \\ c &= -\frac{1}{2}. \end{aligned}$$

Thus, our solution is $u(t) = 3t - \frac{1}{2}$. This answer is correct because it satisfies both the differential equation ($u'(t) = 3$) and the null-space constraint (the mean value of u, which is what we obtain by projecting a function onto the constant functions, is the same as the mean value of β.)

Our second example involves the second derivative operator in one dimension. The second derivative operator is similar to the first derivative, but it has a null space of dimension two. We define our operator, domain, right-hand-side function α, and function β (to be projected onto the null space) as follows:

$$\begin{aligned} \mathcal{D} &= \frac{d^2}{dt^2}, \\ \Omega &= (0, 1), \\ \alpha &= 3, \\ \beta &= t. \end{aligned}$$

From calculus, $\mathcal{D}u = 3$ tells us that $u(t) = \frac{3}{2}t^2 + ct + d$, where d and c are any real numbers. Next, we find $\boldsymbol{V}$ and apply $\boldsymbol{V}^*$ to $u(t)$ and β. The null space of the second derivative operator includes all constant and linear functions. One

basis, which we use here, is $\{1,\, t\}$. Equation (4.6) now gives two additional constraints,

$$\begin{bmatrix} \int_0^1 1 \cdot (3t^2/2 + ct + d)\ \mathrm{dt} \\ \int_0^1 t \cdot (3t^2/2 + ct + d)\ \mathrm{dt} \end{bmatrix} = \begin{bmatrix} \int_0^1 1 \cdot t\ \mathrm{dt} \\ \int_0^1 t \cdot t\ \mathrm{dt} \end{bmatrix}.$$

Simplifying, we obtain

$$\begin{bmatrix} c/2 + d \\ c/3 + d/2 \end{bmatrix} = \begin{bmatrix} 0 \\ -1/24 \end{bmatrix},$$

or

$$\begin{bmatrix} 1/2 & 1 \\ 1/3 & 1/2 \end{bmatrix} \begin{bmatrix} c \\ d \end{bmatrix} = \begin{bmatrix} 0 \\ -1/24 \end{bmatrix}.$$

Solving this system of equations, we find $c = -1/2$ and $d = 1/4$, giving a final solution of $3t^2/2 - t/2 + 1/4$.

The choice of basis for the null space does not affect the final solution. (If u is a solution to (4.4), then $\boldsymbol{V}^* u = \boldsymbol{V}^* \beta$, which implies $\mathcal{P}u = \mathcal{P}\beta$, which in turn implies $\boldsymbol{W}^* u = \boldsymbol{W}^* \beta$ for any null basis $\boldsymbol{W}$ by Lemma 1.) To illustrate this, we will solve the same example problem using a set of orthogonal basis functions $\{1,\, t - 1/2\}$ as a null basis. (These are orthogonal on $(0, 1)$ with respect to the weight function 1.) The system of equations then becomes

$$\begin{aligned} \begin{bmatrix} \int_0^1 1 \cdot (3t^2/2 + ct + d)\ \mathrm{dt} \\ \int_0^1 (t - 1/2) \cdot (3t^2/2 + ct + d)\ \mathrm{dt} \end{bmatrix} &= \begin{bmatrix} \int_0^1 1 \cdot t\ \mathrm{dt} \\ \int_0^1 (t - 1/2) \cdot t\ \mathrm{dt} \end{bmatrix}, \\ \begin{bmatrix} c/2 + d \\ c/12 + 0 \end{bmatrix} &= \begin{bmatrix} 0 \\ -1/24 \end{bmatrix}, \\ \begin{bmatrix} 1/2 & 1 \\ 1/12 & 0 \end{bmatrix} \begin{bmatrix} c \\ d \end{bmatrix} &= \begin{bmatrix} 0 \\ -1/24 \end{bmatrix}, \end{aligned}$$

which looks a little different than the system above but has the same solution, namely, $c = -1/2$, and $d = 1/4$.

The above examples exploit knowledge of the null space and of an analytic solution to our equations. They solve the problem without discretizing in any way. For a more complicated operator we may not be able to find an analytic solution; furthermore, the null space may be infinite-dimensional. In these more general cases, discretization is necessary. Chapter 5 expounds on discretization techniques.

4.3 Relationship to Conventional Methods

The null-space approach to solving differential equations is useful in situations where boundary conditions are not available or not specified. For such problems, it is not possible to compare the null-space approach to conventional approaches because conventional methods require boundary conditions. We can, however, choose some example problems that typically have specified boundary conditions to give an idea of how choosing a β function relates to specifying boundary

conditions.

In one dimension, for problems using the first derivative operator, in selecting a null space constraint function, we specify the average value of the function over the domain. In choosing a null space function for the second derivative operator in one dimension, we specify the average value and the value of the first moment of the solution. These facts are demonstrated by the following examples.

First, we consider

$$\begin{aligned} \mathcal{D} &= \frac{d}{dt}, \\ \Omega &= (0,1), \\ \alpha &= 0. \end{aligned}$$

The analytical solution to this problem without specified boundary conditions is $u(t) = k$, where k is any constant. With the boundary condition $u(0) = c$ applied to our problem, the solution is $u(t) = c$. With the null-space condition, $\mathcal{P}u = \mathcal{P}\beta$ applied to our problem, the solution is $u(t) = b$, where

$$b = \int_0^1 \beta \text{ dt}.$$

(The null space solution is found as demonstrated above in Section 4.2.) It follows that specifying a boundary condition for this problem of $u(0) = b$ isolates the same solution as a null-space constraint function β with an average value of b over the domain.

Similarly, for the problem

$$\begin{aligned} \mathcal{D} &= \frac{d}{dt}, \\ \Omega &= (0,1), \\ \alpha &= a, \end{aligned}$$

the solutions are the same for null constraint function β with average value b and the boundary value $u(0) = b - \frac{a}{2}$.

And, for the problem

$$\begin{aligned} \mathcal{D} &= \frac{d}{dt}, \\ \Omega &= (0,1), \\ \alpha &= a_1 t + a_0, \end{aligned}$$

the solutions are the same for null constraint function β with average value b and the boundary value $u(0) = b - a_1/6 - a_0/2$. This process can be continued for polynomials of higher degree and for other functions as well. Comparisons are more difficult with more complicated differential operators.

Next, consider the problem

$$\begin{aligned} \mathcal{D} &= \frac{d^2}{dt^2}, \\ \Omega &= (0,1), \\ \alpha &= 0. \end{aligned}$$

Here, let

$$b_0 = \int_0^1 \beta \ \mathrm{dt},$$

and

$$b_1 = \int_0^1 \beta t \ \mathrm{dt}.$$

The solutions are the same for null constraint function β and the boundary conditions $u(0) = 4b_0 - 6b_1$ and $u'(0) = 12b_1 - 6b_0$.

In all of these examples, we do not specify a specific β but relate boundary conditions to properties of the β function. This is an important observation because it indicates the nature of the relationship between boundary conditions and the β function. Given a β function as a null-space constraint, we can solve the problem and then from the solution glean Dirichlet boundary values that would give the same solution to the PDE. Given boundary conditions, we can solve the problem and from the solutions we can glean some characteristics of the β function, but we cannot isolate one particular function. In the first example above, β could be b, $2at + b - a$ for any real a, or infinitely many other possible functions, as long as the average value over the interval $(0, 1)$ is b. Knowing a specific value for β is an ill-posed problem because many different functions can have the same projection onto the null space of a given differential operator.

It is difficult to give examples as we did above in two or more dimensions (because of infinite-dimensional null spaces), but comparing the null-space constraint to boundary conditions has limited motivation anyway since we expect null-space methods to be most useful where conventional methods do not apply. The general intuition behind the null-space approach is that when we constrain our solution to have the same null-space component as some other function we choose, we specify something about the shape and position of our solution. We will show how the choice of null function affects the solution values for the electrostatic potential of the exponential density function in Chapter 8.

5 Discretizations

Differential operators operate on smooth functions belonging to an infinite-dimensional function space. Solving a differential equation in that infinite-dimensional function space, however, may be difficult or impossible. Numerical discretization techniques replace the given problem by a finite-dimensional one whose solution in some sense approximates the true solution of the differential equation. There are two basic approaches to such discretization: (1) sample the solution function at a finite number of points in its domain, or (2) represent the solution as a linear combination of a finite set of basis functions. In either case, the accuracy of the resulting approximate solution can usually be made arbitrarily good at the cost of additional computational expense. The various methods available differ in how the sample points or basis functions are chosen and in what sense the resulting computed solution approximates the true solution.

No matter which discretization method we use, the result of discretizing a differential equation using the null-space approach is a linear system of the form

$$\begin{bmatrix} \boldsymbol{D} \\ \hat{\boldsymbol{Z}}^T \end{bmatrix} \boldsymbol{x} = \begin{bmatrix} \boldsymbol{a} \\ \boldsymbol{b} \end{bmatrix}, \tag{5.1}$$

with $\boldsymbol{D}$ an $m \times p$ matrix with $m \leq p$, $\boldsymbol{a}$ an $m \times 1$ vector, $\hat{\boldsymbol{Z}}$ an $p \times k$ matrix with $k = p - \text{rank}(\boldsymbol{D})$, $\boldsymbol{b}$ a $k \times 1$ vector, and our solution $\boldsymbol{x}$ a $p \times 1$ vector. The first set of equations, $\boldsymbol{Dx} = \boldsymbol{a}$, is what a discretization without boundary conditions or any other constraints would produce. The second set of equations comes from the null-space constraint. Together they form an augmented system with a matrix of dimension $(m + k) \times p$. In many cases $k = p - m$, yielding a square matrix. In other cases, when $\boldsymbol{D}$ does not have full row rank, the augmented matrix will be overdetermined in shape. As proved in the following lemmas and theorem, the matrix of equation (5.1) has full column rank in all cases of discrete sampling and linear combination approximations, therefore providing a unique solution in the square case and a unique least squares solution in the overdetermined case.

Lemma 2 *Given a $p \times m$ matrix $\boldsymbol{B}$ with $\text{rank}(\boldsymbol{B}) = m$ and a symmetric positive definite, $p \times p$ matrix $\boldsymbol{C}$, the matrix product*

$$\boldsymbol{B}^T \boldsymbol{C} \boldsymbol{B}$$

is positive definite and thus nonsingular.

Proof: Let $\boldsymbol{x} \in \mathbb{R}^m$ be a nonzero vector. Since $\boldsymbol{B}$ has full column rank, $\boldsymbol{Bx} \neq \boldsymbol{0}$. Let $\boldsymbol{y} = \boldsymbol{Bx}$. Then,

$$\boldsymbol{y}^T\boldsymbol{Cy} > 0$$

by positive definiteness of $\boldsymbol{C}$. It follows that

$$\boldsymbol{x}^T\boldsymbol{B}^T\boldsymbol{CBx} > 0$$

for all nonzero $\boldsymbol{x}$, implying that $\boldsymbol{B}^T\boldsymbol{CB}$ is positive definite. □

Lemma 3 *Given an $m \times p$ matrix $\boldsymbol{A}$, with $m \leq p$ and $\text{rank}(\boldsymbol{A}) = k$ with $k < p$, $\boldsymbol{B}$ a basis for the null space of $\boldsymbol{A}$, and a symmetric, positive definite, $p \times p$ matrix $\boldsymbol{C}$,*

$$\boldsymbol{L} = \begin{bmatrix} \boldsymbol{A} \\ \boldsymbol{B}^T\boldsymbol{C} \end{bmatrix}$$

has rank p (i.e., full column rank).

Proof: The matrix $\boldsymbol{B}$ is a basis for $N(\boldsymbol{A})$, so it is $p \times (p-k)$. Matrix $\boldsymbol{L}$, therefore, is $(m+p-k) \times p$. By $\text{rank}(\boldsymbol{A}) = k$ and $m \leq p$, we know $k \leq m$, which implies $m + p - k \geq p$. Since $\boldsymbol{L}$ is square or overdetermined in shape, it suffices to show that the null space of $\boldsymbol{L}$ is trivial.

Let $\boldsymbol{x}$ be a vector in the null space of $\boldsymbol{L}$, that is,

$$\boldsymbol{Lx} = 0.$$

This implies

$$\boldsymbol{Ax} = 0,$$

and that $\boldsymbol{x}$ can be written as a linear combination of null vectors of $\boldsymbol{A}$,

$$\boldsymbol{x} = \boldsymbol{By}, \tag{5.2}$$

for some vector $\boldsymbol{y}$. $\boldsymbol{Lx} = 0$ further implies that

$$\boldsymbol{B}^T\boldsymbol{Cx} = 0.$$

Substituting for $\boldsymbol{x}$ using (5.2), we obtain

$$\boldsymbol{B}^T\boldsymbol{CBy} = 0,$$

which implies $y = 0$ because $\boldsymbol{B}^T\boldsymbol{CB}$ is nonsingular by Lemma 2. This, in turn, implies that $\boldsymbol{x} = 0$ by (5.2). The only vector in the null space of $\boldsymbol{L}$, then, is the zero vector. Therefore $\boldsymbol{L}$ has a trivial null space and full column rank. □

Theorem 1 *The linear system (5.1), derived from the null-space approach for*

solving differential equations, has full column rank for discrete sampling and linear combination approximations.

Proof: As further explained in the following sections, in the case of a discrete solution, such as that obtained by a finite difference approximation with $u \approx \hat{\boldsymbol{u}} = \boldsymbol{x}$, $\hat{\boldsymbol{Z}}^T = \boldsymbol{Z}^T$, and in the case of an approximation of u as a linear combination of basis functions, $u \approx \hat{u} = \boldsymbol{\Phi}\boldsymbol{x}$, $\hat{\boldsymbol{Z}}^T = \boldsymbol{Z}^T\boldsymbol{\Phi}^*\boldsymbol{\Phi}$. We consider these cases separately.

Case 1: $\hat{\boldsymbol{Z}}^T = \boldsymbol{Z}^T$

Suppose the $m \times p$ matrix $\boldsymbol{D}$ has rank k. Then $\boldsymbol{Z}$ has $p - k$ linearly independent columns, each orthogonal to the rows of $\boldsymbol{D}$. Thus, $\boldsymbol{D}$ and $\boldsymbol{Z}^T$ have mutually orthogonal row spaces, and

$$\operatorname{rank}\left(\begin{bmatrix} \boldsymbol{D} \\ \hat{\boldsymbol{Z}}^T \end{bmatrix}\right) = k + (p - k) = p.$$

Case 2: $\hat{\boldsymbol{Z}}^T = \boldsymbol{Z}^T\boldsymbol{\Phi}^*\boldsymbol{\Phi}$. By construction of the problem, $\boldsymbol{\Phi}$ contains linearly independent functions, so $\boldsymbol{\Phi}^*\boldsymbol{\Phi}$ is symmetric and positive definite. Thus, the problem matrix has full column rank by Lemma 3.

□

5.1 Finite Difference Methods

Finite difference methods sample the solution function on a discrete mesh of points distributed throughout the problem domain. The continuous solution function is replaced by a p-dimensional vector whose entries represent sample values of the solution function at the mesh points, and the differential operator is replaced by a finite-difference quotient, resulting in a system of algebraic equations to be solved for the approximate solution at each of the mesh points. As the resolution of the mesh is refined, the approximate solution values at the mesh points converge, under suitable conditions, to the corresponding values of the true solution of the original continuous problem.

We illustrate the finite difference approach first for Poisson's equation in one dimension, say on the interval $(0, 1)$:

$$\Delta u = u'' = \alpha, \tag{5.3}$$

where $\alpha\colon (0,1) \to \mathbb{R}$ is known and $u\colon (0,1) \to \mathbb{R}$ is the unknown solution function to be determined. Let $\boldsymbol{t} \in \mathbb{R}^n$ be a vector whose entries represent a discrete mesh of points in $(0, 1)$, with

$$t_i = (i-1)h, \qquad i = 1, \ldots, n, \tag{5.4}$$

where $h = 1/(n-1)$. Using the second-order, centered finite difference approx-

imation to the second derivative,

$$u''(t_i) \approx \frac{u(t_{i+1}) - 2u(t_i) + u(t_{i-1})}{h^2}, \tag{5.5}$$

we obtain a system of algebraic equations of the form

$$\boldsymbol{D}\hat{\boldsymbol{u}} = \frac{1}{h^2}\begin{bmatrix} 1 & -2 & 1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & -2 & 1 & 0 & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & 1 & -2 & 1 & 0 \\ 0 & \cdots & 0 & 0 & 1 & -2 & 1 \end{bmatrix} \begin{bmatrix} \hat{u}_1 \\ \hat{u}_2 \\ \vdots \\ \hat{u}_{n-1} \\ \hat{u}_n \end{bmatrix} = \begin{bmatrix} a_2 \\ a_3 \\ \vdots \\ a_{n-2} \\ a_{n-1} \end{bmatrix} = \boldsymbol{a}, \tag{5.6}$$

where $\hat{u}_i$ approximates $u(t_i)$, $i = 1, \ldots, n$, and $a_i = \alpha(t_i)$, $i = 2, \ldots, n-1$. Observe that the $(n-2) \times n$ system (5.6) is underdetermined, owing to the fact that the centered finite difference stencil (5.5) cannot be applied at the end points of the interval. Thus, $\boldsymbol{D}$ has a nontrivial null space, and hence the system (5.6) cannot have a unique solution, because for any solution $\hat{\boldsymbol{u}}$, $\hat{\boldsymbol{u}} + \boldsymbol{v}$ is also a solution, where $\boldsymbol{v} \in N(\boldsymbol{D})$. In particular, $\text{rank}(\boldsymbol{D}) = n - 2$, and the two-dimensional null space $N(\boldsymbol{D})$ is spanned by the columns of the $n \times 2$ matrix

$$\boldsymbol{Z} = \begin{bmatrix} 1 & 1 \\ 1 & 2 \\ 1 & 3 \\ \vdots & \vdots \\ 1 & n \end{bmatrix}, \tag{5.7}$$

corresponding exactly to the two-dimensional null space of the second-derivative operator in one dimension, which annihilates all constant and linear functions.

Conventionally, a particular solution to (5.6) is singled out by imposing boundary conditions at the end points, which in this context means specifying the first and last entries of $\boldsymbol{u}$. Thus, we solve the $n \times n$ augmented system

$$\begin{bmatrix} \boldsymbol{D} \\ \boldsymbol{e}_1^T \\ \boldsymbol{e}_n^T \end{bmatrix} \hat{\boldsymbol{u}} = \begin{bmatrix} \boldsymbol{a} \\ \gamma_1 \\ \gamma_2 \end{bmatrix}, \tag{5.8}$$

where γ_1 and γ_2 are the desired values for $\hat{u}_1$ and $\hat{u}_n$. This square, nonsingular linear system has a unique solution $\hat{\boldsymbol{u}}$ that satisfies the boundary conditions as well as (5.6). Equivalently, since $\hat{u}_1$ and $\hat{u}_n$ are fully determined by the boundary conditions, they can be taken to the right-hand-side of equation (5.6), effectively removing the first and last columns of matrix $\boldsymbol{D}$ and resulting in an $(n-2) \times (n-2)$ system to be solved for $\hat{u}_2, \ldots, \hat{u}_{n-1}$.

Rather than imposing boundary conditions, we could instead specify the desired null-space component of the solution explicitly. Formulation of the discretized system proceeds in the same way as the formulation of the continuous

problem described in Chapter 4. The orthogonal projector onto $N(\boldsymbol{D})$ is given by the $n \times n$ symmetric idempotent matrix

$$\boldsymbol{P} = \boldsymbol{Z}(\boldsymbol{Z}^T\boldsymbol{Z})^{-1}\boldsymbol{Z}^T. \tag{5.9}$$

To force the solution to (5.6) to have a prescribed projection onto $N(\boldsymbol{D})$, we therefore solve the $(2n-2)\times n$ overdetermined but compatible augmented system

$$\begin{bmatrix} \boldsymbol{D} \\ \boldsymbol{P} \end{bmatrix} \hat{\boldsymbol{u}} = \begin{bmatrix} \boldsymbol{a} \\ \boldsymbol{Z\gamma} \end{bmatrix}, \tag{5.10}$$

where $\boldsymbol{\gamma} \in \mathbb{R}^2$. But the condition $\boldsymbol{P}\hat{\boldsymbol{u}} = \boldsymbol{Z\gamma}$ is necessarily satisfied if $\boldsymbol{Z}^T\hat{\boldsymbol{u}} = \boldsymbol{Z}^T\boldsymbol{Z\gamma}$, and hence it can be replaced by the latter condition, yielding the $n \times n$, nonsingular augmented system

$$\begin{bmatrix} \boldsymbol{D} \\ \boldsymbol{Z}^T \end{bmatrix} \hat{\boldsymbol{u}} = \begin{bmatrix} \boldsymbol{a} \\ \boldsymbol{Z}^T\boldsymbol{Z\gamma} \end{bmatrix}, \tag{5.11}$$

whose solution $\hat{\boldsymbol{u}}$ satisfies (5.6) and also has the desired null-space component prescribed by $\boldsymbol{\gamma}$. Note that if the basis for $N(\boldsymbol{D})$ is orthonormal, i.e., $\boldsymbol{Z}^T\boldsymbol{Z} = \boldsymbol{I}$, then the augmented system (5.11) simplifies to

$$\begin{bmatrix} \boldsymbol{D} \\ \boldsymbol{Z}^T \end{bmatrix} \hat{\boldsymbol{u}} = \begin{bmatrix} \boldsymbol{a} \\ \boldsymbol{\gamma} \end{bmatrix}. \tag{5.12}$$

The null-space constraint is sometimes given in terms of a function β rather than as null space components. In this case, (5.10) becomes

$$\begin{bmatrix} \boldsymbol{D} \\ \boldsymbol{P} \end{bmatrix} \hat{\boldsymbol{u}} = \begin{bmatrix} \boldsymbol{a} \\ \boldsymbol{Pc} \end{bmatrix}, \tag{5.13}$$

where $\boldsymbol{c} \in \mathbb{R}^n$ is a discretization of β, with $c_i = \beta(t_i)$. This second condition, in turn, is necessarily satisfied if $\boldsymbol{Z}^T\hat{\boldsymbol{u}} = \boldsymbol{Z}^T\boldsymbol{c}$ providing us another $n \times n$, nonsingular augmented system,

$$\begin{bmatrix} \boldsymbol{D} \\ \boldsymbol{Z}^T \end{bmatrix} \hat{\boldsymbol{u}} = \begin{bmatrix} \boldsymbol{a} \\ \boldsymbol{Z}^T\boldsymbol{c} \end{bmatrix}, \tag{5.14}$$

which is useful when β is specified rather than values for γ. This correlates to our general problem (5.1) with $\hat{\boldsymbol{Z}}^T = \boldsymbol{Z}^T$ and $\boldsymbol{b} = \boldsymbol{Z}^T\boldsymbol{c}$.

As a specific example, we solve Poisson's equation in one dimension with $\alpha = 3$ and $\beta = t$ on the interval $(0, 1)$. That is,

$$\begin{aligned} \mathcal{D} &= \tfrac{d^2}{dt^2}, \\ \Omega &= (0,1), \\ \alpha &= 3, \\ \beta &= t. \end{aligned} \tag{5.15}$$

For comparison, we solved the same problem analytically in Section 4.2; the exact solution is $u(t) = 3t^2/2 - t/2 + 1/4$. We will solve the problem with three mesh points, yielding an initial problem matrix with only one row:

$$\boldsymbol{D}\hat{\boldsymbol{u}} = \begin{bmatrix} 4 & -8 & 4 \end{bmatrix} \begin{bmatrix} \hat{u}_1 \\ \hat{u}_2 \\ \hat{u}_3 \end{bmatrix} = \begin{bmatrix} 3 \end{bmatrix} = \boldsymbol{a}.$$

Our null basis is spanned by

$$\boldsymbol{Z} = \begin{bmatrix} 1 & 1 \\ 1 & 2 \\ 1 & 3 \end{bmatrix},$$

and our discretized β, which is necessary for the right-hand side, is

$$\boldsymbol{c} = \begin{bmatrix} 0 \\ 1/2 \\ 1 \end{bmatrix}.$$

Putting the pieces together according to equation (5.14), we obtain the complete system

$$\begin{bmatrix} \boldsymbol{D} \\ \boldsymbol{Z}^T \end{bmatrix} \hat{\boldsymbol{u}} = \begin{bmatrix} 4 & -8 & 4 \\ 1 & 1 & 1 \\ 1 & 2 & 3 \end{bmatrix} \begin{bmatrix} \hat{u}_1 \\ \hat{u}_2 \\ \hat{u}_3 \end{bmatrix} = \begin{bmatrix} 3 \\ 3/2 \\ 4 \end{bmatrix} = \begin{bmatrix} \boldsymbol{a} \\ \boldsymbol{Z}^T\boldsymbol{c} \end{bmatrix},$$

which has solution $\hat{\boldsymbol{u}} = \begin{bmatrix} 1/8 & 1/4 & 9/8 \end{bmatrix}^T$. This solution vector does not match the exact solution $u(t) = 3t^2/2 - t/2 + 1/4$ at the mesh points. In fact, the error is

$$\begin{bmatrix} 1/8 \\ 1/4 \\ 9/8 \end{bmatrix} - \begin{bmatrix} 1/4 \\ 3/8 \\ 5/4 \end{bmatrix} = \begin{bmatrix} -1/8 \\ -1/8 \\ -1/8 \end{bmatrix}.$$

The solution and the finite difference approximation are graphed in Figure (5.1). A larger value of n would yield a better approximate answer.

We next consider Poisson's equation in two dimensions, say on the unit square $(0,1) \times (0,1)$:

$$\Delta u = u_{ss} + u_{tt} = \alpha(s,t), \tag{5.16}$$

where α and u are now real-valued functions of two variables. Using $\boldsymbol{t}$ as defined in (5.4), we introduce an $n \times n$ mesh of points (t_i, t_j), $i, j = 1, \ldots n$, where for simplicity we use the same mesh spacing h in both dimensions. Applying the second-order, centered finite difference quotient (5.5) to approximate the second partial derivatives with respect to s and t at each of the $(n-2)^2$ interior mesh points, we obtain an $(n-2)^2 \times n^2$ block-tridiagonal system of equations of the form

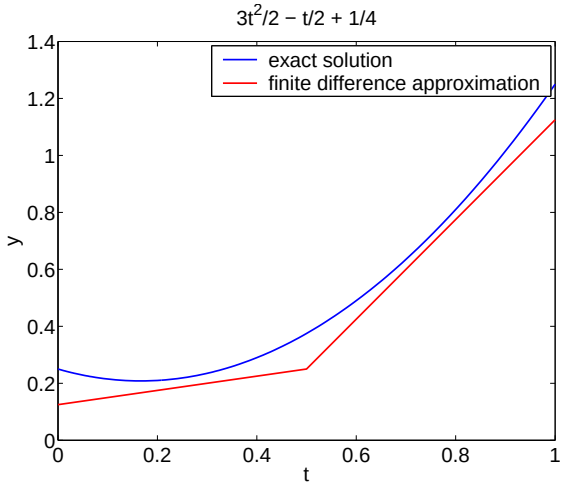

Figure 5.1: Exact and finite difference solutions to 1-D Poisson example

$$\boldsymbol{D}\hat{\boldsymbol{u}} = \frac{1}{h^2}\begin{bmatrix} \boldsymbol{D}_1 & \boldsymbol{D}_2 & \boldsymbol{D}_1 & \boldsymbol{O} & \boldsymbol{O} & \cdots & \boldsymbol{O} \\ \boldsymbol{O} & \boldsymbol{D}_1 & \boldsymbol{D}_2 & \boldsymbol{D}_1 & \boldsymbol{O} & \cdots & \boldsymbol{O} \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ \boldsymbol{O} & \cdots & \boldsymbol{O} & \boldsymbol{D}_1 & \boldsymbol{D}_2 & \boldsymbol{D}_1 & \boldsymbol{O} \\ \boldsymbol{O} & \cdots & \boldsymbol{O} & \boldsymbol{O} & \boldsymbol{D}_1 & \boldsymbol{D}_2 & \boldsymbol{D}_1 \end{bmatrix} \begin{bmatrix} \hat{u}_1 \\ \hat{u}_2 \\ \vdots \\ \hat{u}_{n^2-1} \\ \hat{u}_{n^2} \end{bmatrix} = \boldsymbol{a}, \tag{5.17}$$

where the nonzero $(n-2) \times n$ blocks are given by

$$\boldsymbol{D}_1 = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 1 & 0 & 0 & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & 0 & 1 & 0 & 0 \\ 0 & \cdots & 0 & 0 & 0 & 1 & 0 \end{bmatrix} \tag{5.18}$$

and

$$\boldsymbol{D}_2 = \begin{bmatrix} 1 & -4 & 1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & -4 & 1 & 0 & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & 1 & -4 & 1 & 0 \\ 0 & \cdots & 0 & 0 & 1 & -4 & 1 \end{bmatrix}, \tag{5.19}$$

$\hat{u}_{i+(j-1)n}$ approximates $u(t_i, t_j)$ at a given mesh point, and the entries of $\boldsymbol{a} \in \mathbb{R}^{(n-2)^2}$ are evaluated at the $(n-2)^2$ corresponding interior mesh points.

Again, the $(n-2)^2 \times n^2$ system (5.17) is underdetermined and therefore has a nontrivial null space and cannot have a unique solution. In particular, $\text{rank}(\boldsymbol{D}) = (n-2)^2$, so $N(\boldsymbol{D})$ is of dimension $n^2 - (n-2)^2 = 4(n-1)$. Conventionally, a particular solution to (5.17) is singled out by imposing boundary conditions at the $4(n-1)$ mesh points on the boundary, effectively resulting in

a square, nonsingular system of dimension $(n-2)^2$ to solve for the approximate solution values at the $(n-2)^2$ interior mesh points.

Alternatively, given an $n^2 \times 4(n-1)$ matrix $\boldsymbol{Z}$ whose columns form a basis for $N(\boldsymbol{D})$, we could specify the null space component of the solution to (5.17) explicitly, as we have seen before. A new twist arises here, however: the null space of the Laplacian operator Δ in two dimensions is infinite dimensional, in that, as can easily be confirmed by differentiation, the Laplacian annihilates not only constant and bilinear functions, but more general polynomials in s and t, such as $a + bs + ct + dst + e(s^2 - t^2)$, where a, b, c, d, e are arbitrary constants, as well as products such as $\sin(\lambda s)\exp(\lambda t)$ or $\sin(\lambda s)\sinh(\lambda t)$, where λ is an arbitrary constant, and indeed any *harmonic* function, of which there is an infinite variety (see [10, 20, 54], for example, for more information on harmonic functions). Thus, the $4(n-1)$-dimensional subspace spanned by the columns of the matrix $\boldsymbol{Z}$ provides only an approximation to the true null space of the Laplacian. In any case, we can still use $\boldsymbol{Z}$ to specify the desired null-space component of the solution to (5.17), which itself provides only a discrete approximation to the true solution of (5.16).

It is instructive to compare the situation just described for the null-space approach to the conventional approach of imposing boundary conditions. Note that the choice of boundary data for the continuous problem (5.16) involves an infinite number of degrees of freedom, since boundary values must be specified on the continuum of all boundary points, but the discrete problem (5.17) allows only $4(n-1)$ degrees of freedom, since boundary values are specified only at $4(n-1)$ mesh points on the boundary. Thus, the conventional boundary value approach has no more "expressive power" than the null-space approach, and both enjoy correspondingly improved accuracy as the mesh is refined.

To demonstrate the convergence of the null-space approach approximations to the exact solution as the mesh is refined, we solve the 2-D Poisson problem with

$$\begin{aligned} \mathcal{D} &= \Delta = \tfrac{\partial^2}{\partial s^2} + \tfrac{\partial^2}{\partial t^2}, \\ \Omega &= (0,1) \times (0,1), \\ \alpha &= 12s^2t^2 + 2t^4, \\ \beta &= s^2t^4, \end{aligned} \tag{5.20}$$

for several values of n. Since our choice of null-space function β satisfies the equation, we know that it is the exact solution to the problem. That is, $u = \beta$. We solved the problem in MATLAB using the `null()` function to obtain a null basis, and we use the maximum norm over the nodal values as a measure of error. Figure 5.2 shows the error in the finite difference approximations as n increases based on data for $n = 5$, 15, 25, and 35. The dashed lines in the plot show linear and quadratic convergence curves. From the graph, it is clear that this method demonstrates quadratic convergence in n. The total number of nodes in the problem is n^2, however, so the method exhibits linear convergence in the total number of nodes.

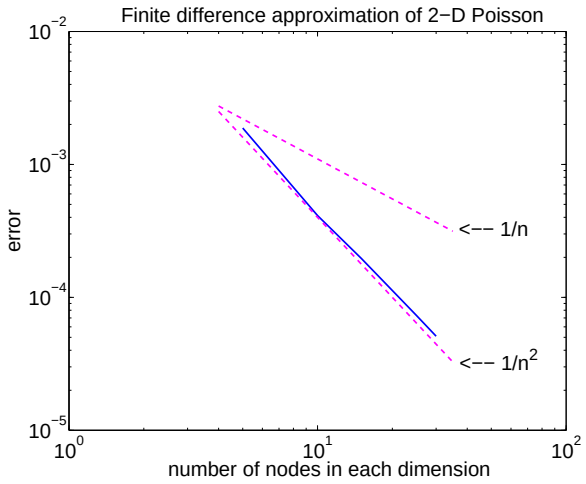

Figure 5.2: Error in finite difference approximation for 2-D Poisson example

5.2 Galerkin Methods

Galerkin methods approximate the solution of a differential equation by a linear combination of a finite set of linearly independent basis functions—often called *trial* functions in this context—defined on the problem domain. The computational problem, then, is to determine the coefficients of that linear combination. The trial functions chosen may have local support, as in finite element methods, or global support, as in spectral methods. In either case, most of the trial functions are usually chosen so that they vanish on the boundary of the problem domain Ω (we will refer to these as the *homogeneous* basis functions).

The coefficients of the linear combination of trial functions approximating the solution are determined by forcing it to satisfy the differential equation in an averaged sense. The linear combination is substituted into the differential equation, and the resulting residual is forced to be orthogonal to each of a second set of functions, called *test* functions, which results in a system of equations to be solved for the coefficients of the linear combination of trial functions. In the Galerkin approach, the test functions are chosen to be the homogeneous trial functions, and the remaining, inhomogeneous trial functions are used to satisfy whatever boundary conditions may apply.

The smoothness requirements on the trial functions can often be reduced through integration by parts (or applying the Divergence Theorem in higher dimensions), which may also accrue other advantages, such as symmetry. Although the resulting approximate solution may lack the differentiability of the true solution, nevertheless it can still approximate the true solution arbitrarily closely as the number of basis functions increases.

We illustrate the Galerkin method first for the Poisson problem (5.3) in one dimension on the interval $(0, 1)$. We approximate the solution to (5.3) by a

linear combination of a set trial functions $\boldsymbol{\Phi} = [\,\phi_1 \quad \phi_2 \quad \cdots \quad \phi_n\,] \subseteq L_2(0,1)$, so that

$$u \approx \boldsymbol{\Phi}\boldsymbol{x}, \tag{5.21}$$

where $\boldsymbol{x} \in \mathbb{R}^n$ is to be determined. We will take the set of test functions to be the homogeneous trial functions, which we denote by $\boldsymbol{\Phi}_0$, and require that (5.3) be satisfied in the *weak* sense,

$$\boldsymbol{\Phi}_0^* u'' = \boldsymbol{\Phi}_0^* \alpha. \tag{5.22}$$

Integrating the left side of (5.22) by parts and using the homogeneity of the test functions, we obtain

$$(-\boldsymbol{\Phi}_0')^* u' = \boldsymbol{\Phi}_0^* \alpha. \tag{5.23}$$

Differentiating the approximation (5.21) and substituting into (5.23), we obtain the $m \times n$ system of linear equations

$$\boldsymbol{D}\boldsymbol{x} = \boldsymbol{a} \tag{5.24}$$

where $\boldsymbol{D} = [(-\boldsymbol{\Phi}_0')^* \boldsymbol{\Phi}']$, $\boldsymbol{a} = \boldsymbol{\Phi}_0^* \alpha$, and where m ($\leq n$) is the number of homogeneous trial functions.

Again, we have arrived at an underdetermined linear system with a nontrivial null space and thus a nonunique solution. While the matrix $\boldsymbol{D}$ and the vector $\boldsymbol{a}$ will vary from problem to problem depending on the differential operator, all lead to an underdetermined linear system (by rank, if not by shape). Conventionally, a specific solution is singled out by imposing boundary conditions, which in this context involves the inhomogeneous trial functions. Alternatively, we could specify the null-space component of the solution. Note, however, that the approximate solution is now a continuous function rather than a discrete vector, and the null space of the discretized operator is similarly a function space rather than a discrete vector space. Hence, we need to project the approximate solution onto a subspace of functions, as in Section 2.4.

Let $\boldsymbol{Z}$ be an $n \times k$ matrix whose columns form a basis for $N(\boldsymbol{D})$, the null space of $\boldsymbol{D}$ (i.e., $\boldsymbol{D}\boldsymbol{Z} = \boldsymbol{0}$). We expect that the discretized operator $\boldsymbol{D}$ will inherit a nontrivial null space (i.e., $k \geq 1$) from the continuous operator $\mathcal{D}$. For $j = 1, ..., k$ we define *null functions* $v_j \in C_1(\Omega)$ by

$$v_j(t) = \sum_{i=1}^{n} z_{ij}\, \phi_i(t). \tag{5.25}$$

The entire set of null functions is $\boldsymbol{V} = \boldsymbol{\Phi}\boldsymbol{Z}$. Section 5.4 provides a proof that these functions are indeed in $N(\mathcal{D})$. We know that the null functions are linearly independent because $\boldsymbol{\Phi}$ contains linearly independent functions and $\boldsymbol{Z}$ is a null basis, which implies it has linearly independent rows.

It follows that $\boldsymbol{V}^* = \boldsymbol{Z}^T\boldsymbol{\Phi}^*$. The second set of equations from (4.4) becomes

$$\boldsymbol{Z}^T\boldsymbol{\Phi}^* u = \boldsymbol{Z}^T\boldsymbol{\Phi}^*\beta,$$

and substituting $\hat{u}$ for u, we obtain

$$\boldsymbol{Z}^T\boldsymbol{\Phi}^*\boldsymbol{\Phi}\boldsymbol{x} = \boldsymbol{Z}^T\boldsymbol{\Phi}^*\beta. \tag{5.26}$$

The complete problem thus becomes

$$\begin{bmatrix} \boldsymbol{D} \\ \boldsymbol{Z}^T\boldsymbol{\Phi}^*\boldsymbol{\Phi} \end{bmatrix} \boldsymbol{x} = \begin{bmatrix} \boldsymbol{a} \\ \boldsymbol{Z}^T\boldsymbol{\Phi}^*\beta \end{bmatrix}, \tag{5.27}$$

which matches the form (5.1) with $\hat{\boldsymbol{Z}}^T = \boldsymbol{Z}^T\boldsymbol{\Phi}^*\boldsymbol{\Phi}$ and $\boldsymbol{b} = \boldsymbol{Z}^T\boldsymbol{\Phi}^*\beta$.

We now illustrate the Galerkin method by solving the Poisson equation in one dimension. We will use the differential operator, α, β, and Ω given in Equation (5.15). For basis functions, we introduce mesh points at 0, 1/2, and 1 and define the piecewise linear "hat" functions on $(0,1)$ by

$$\begin{aligned}
\phi_1(t) &= \begin{cases} 1-2t, & 0 \le t \le 1/2 \\ 0, & 1/2 < t \le 1 \end{cases}, \\
\phi_2(t) &= \begin{cases} 2t, & 0 \le t \le 1/2 \\ 2-2t, & 1/2 < t \le 1 \end{cases}, \\
\phi_3(t) &= \begin{cases} 0, & 0 \le t \le 1/2 \\ 2t-1, & 1/2 < t \le 1 \end{cases}.
\end{aligned}$$

Technically, these basis functions are not in $C_1(0,1)$, since they are not differentiable at the mesh points, but they and their piecewise constant derivatives,

$$\begin{aligned}
\phi_1'(t) &= \begin{cases} -2, & 0 \le t \le 1/2 \\ 0, & 1/2 < t \le 1 \end{cases}, \\
\phi_2'(t) &= \begin{cases} 2, & 0 \le t \le 1/2 \\ -2, & 1/2 < t \le 1 \end{cases}, \\
\phi_3'(t) &= \begin{cases} 0, & 0 \le t \le 1/2 \\ 2, & 1/2 < t \le 1 \end{cases},
\end{aligned}$$

are integrable, so there is no formal impediment to their use for our purposes.

Only one of our trial functions is homogenous, namely $\phi_2(t)$, so $\boldsymbol{\Phi}_0 = [\phi_2]$. Substituting into (5.24) and evaluating the simple integrals analytically, we obtain

$$\boldsymbol{D}\boldsymbol{x} = \begin{bmatrix} -2 & 4 & -2 \end{bmatrix} \boldsymbol{x} = \begin{bmatrix} 3/2 \end{bmatrix} = \boldsymbol{a}.$$

Since the row sums of $\boldsymbol{D}$ are zero, one null vector of the matrix is given by $\begin{bmatrix} 1 & 1 & 1 \end{bmatrix}^T$; a second null vector is given by $\begin{bmatrix} 1 & 2 & 3 \end{bmatrix}^T$. Thus from (5.25) we have $v_1(t) = \phi_1(t)+\phi_2(t)+\phi_3(t) \equiv 1$, and $v_2(t) = \phi_1(t)+2\cdot\phi_2(t)+3\cdot\phi_3(t) \equiv t+1$,

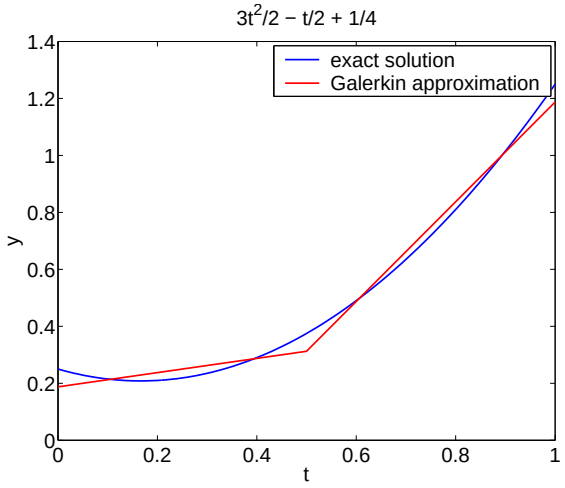

Figure 5.3: Exact and Galerkin method solutions to 1-D Poisson example

which certainly lie in $N(\mathcal{D})$. Evaluating the integrals in (5.26) analytically, we obtain

$$\boldsymbol{Z}^T(\boldsymbol{\Phi}^*\boldsymbol{\Phi}) = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 2 & 3 \end{bmatrix} \begin{bmatrix} 1/6 & 1/12 & 0 \\ 1/12 & 1/3 & 1/12 \\ 0 & 1/12 & 1/6 \end{bmatrix} = \begin{bmatrix} 1/4 & 1/2 & 1/4 \\ 1/3 & 1 & 2/3 \end{bmatrix}$$

for the lower submatrix on the left and

$$\boldsymbol{Z}^T(\boldsymbol{\Phi}^*\beta) = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 2 & 3 \end{bmatrix} \begin{bmatrix} 1/24 \\ 1/4 \\ 5/24 \end{bmatrix} = \begin{bmatrix} 1/2 \\ 7/6 \end{bmatrix}$$

for the lower vector on the right. Altogether, our system is

$$\begin{bmatrix} \boldsymbol{D} \\ \boldsymbol{Z}^T\boldsymbol{\Phi}^*\boldsymbol{\Phi} \end{bmatrix} \boldsymbol{x} = \begin{bmatrix} 2 & -4 & 2 \\ 1/4 & 1/2 & 1/4 \\ 1/3 & 1 & 2/3 \end{bmatrix} \boldsymbol{x} = \begin{bmatrix} 3/2 \\ 1/2 \\ 7/6 \end{bmatrix} = \begin{bmatrix} \boldsymbol{a} \\ \boldsymbol{Z}^T\boldsymbol{\Phi}^*\beta \end{bmatrix},$$

which has unique solution

$$\boldsymbol{x} = [3/16 \quad 5/16 \quad 19/16]^T = [0.1875 \quad 0.3125 \quad 1.1875]^T.$$

Using this solution vector as coefficients in (5.21), we obtain the continuous solution $\hat{u} = 1/16 \cdot (3\phi_1 + 5\phi_2 + 19\phi_3)$, as shown in Figure 5.3.

Next, we demonstrate the Galerkin method for a 2-D Poisson example, specifically the problem given in Equation (5.20), which has the solution $u(s,t) = s^2t^4$. We uniformly discretize the unit square with n grid points in each dimension, producing $(n-1)^2$ squares (elements) over the domain. As basis functions, we use piecewise "tent" functions, one for each intersection (grid

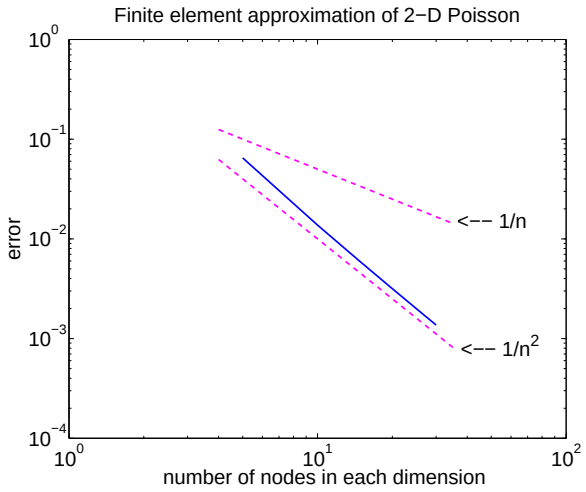

Figure 5.4: Error in Galerkin fin. el. approximation for 2-D Poisson example

point) on the grid. Each of these basis functions has a value of one at its corresponding grid point and is zero elsewhere; they are piecewise defined over the elements of the domain and can be formed as the products of piecewise "hat" functions defined on the line. As an example, let $\psi_1(t)$ be the linear function such that $\psi_1(a) = 1$ and $\psi_1(b) = 0$ and let $\psi_2(t)$ be the linear function such that $\psi_2(a) = 0$ and $\psi_2(b) = 1$. Then, on the two-dimensional domain $(a, b) \times (a, b)$, the pieces of basis functions would be $\phi_{ij}(t) = \psi_i(t)\psi_j(t)$. These are standard bilinear finite elements.

As in the previous finite difference example, we plot the error in the Galerkin finite element approximation for various values of n, as shown in Figure 5.4. The Galerkin method shows quadratic convergence in n and linear convergence in the number of nodes.

5.3 Collocation Methods

Collocation methods represent the solution function as a linear combination of a finite set of basis functions defined on the problem domain,

$$u \approx \hat{u} = \boldsymbol{\Phi x} \tag{5.28}$$

as in Galerkin methods above. The collocation approach similarly yields a system of equations to be solved for the coefficients of the linear combination though in this case, the system is formed by forcing the residual of the differential equation to be zero at each of a finite set of mesh points (called collocation points in this context). With this constraint, our solution function exactly satisfies the differential equation at a certain number of points but may not match the exact solution value at any of those points, nor the differential equation between those

points. Accuracy of the solution is related to the number of basis functions used and the corresponding number of collocation points.

The linear system of equations is formed by substituting the approximate solution (5.28) into the differential equation and evaluating it at each collocation point, producing a linear system with dimension [number of collocation points] by [number of basis functions]. The number of collocation points on the domain is chosen to match the number of basis functions (providing the same number of equations as unknowns), and the positions of the collocation points are generally chosen to include the boundary of the domain. With the conventional approach, the collocation points on the boundary are constrained to satisfy the boundary conditions. In the context of null-space methods, however, boundary conditions are unspecified, and we use only interior collocation points to form our system, yielding a linear system with dimension [number of interior collocation points] by [number of basis functions], which is underdetermined by construction.

The null vectors and null space constraint follow the same way here as they do in the Galerkin approach. Null functions are defined as linear combinations of null vectors of the problem matrix and basis functions. The complete problem form is the same as that given in Equation 5.27:

$$\begin{bmatrix} \boldsymbol{D} \\ \boldsymbol{Z}^T \boldsymbol{\Phi}^* \boldsymbol{\Phi} \end{bmatrix} \boldsymbol{x} = \begin{bmatrix} \boldsymbol{a} \\ \boldsymbol{Z}^T \boldsymbol{\Phi}^* \beta \end{bmatrix}, \tag{5.29}$$

where $\boldsymbol{\Phi}$ contains the basis functions of the linear combination approximation (i.e., from (5.28).)

We now demonstrate a collocation method on a 1-D Poisson equation, namely Equation (5.15). Rather than using the piecewise linear basis functions from the Galerkin example above, which will all be annihilated in our equation, we use the first three monomials as our basis functions, 1, t, and t^2.

We take the one interior collocation point to be the point $t = 1/2$. Our equations will have the form

$$\begin{aligned} \frac{d^2}{dt^2}\hat{u} &= 3, \\ \frac{d^2}{dt^2}\left(\sum_i x_i \phi_i\right) &= 3, \\ \sum_i x_i \phi_i'' &= 3. \end{aligned} \tag{5.30}$$

Taking the second derivative of our basis functions, we have

$$\begin{aligned} \phi_1'' &= 0, \\ \phi_2'' &= 0, \\ \phi_3'' &= 2. \end{aligned}$$

Putting these equations into (5.30) above and evaluating the result at $t = 1/2$,

we have our initial system,

$$\boldsymbol{D}\boldsymbol{x} = \begin{bmatrix} 0 & 0 & 2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = [3] = \boldsymbol{a}.$$

A null basis for this problem matrix is

$$\boldsymbol{Z} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix}.$$

These null vectors correspond to null functions $v_1 = 1 \cdot \phi_1 = 1$ and $v_2 = 1 \cdot \phi_2 = t$, which are clearly in the null space of the Laplacian operator. Evaluating the integrals in (5.27) analytically, we obtain

$$\boldsymbol{Z}^T(\boldsymbol{\Phi}^*\boldsymbol{\Phi}) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} 1 & 1/2 & 1/3 \\ 1/2 & 1/3 & 1/4 \\ 1/3 & 1/4 & 1/5 \end{bmatrix} = \begin{bmatrix} 1 & 1/2 & 1/3 \\ 1/2 & 1/3 & 1/4 \end{bmatrix}$$

for the lower submatrix on the left and

$$\boldsymbol{Z}^T(\boldsymbol{\Phi}^*\beta) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} 1/2 \\ 1/3 \\ 1/4 \end{bmatrix} = \begin{bmatrix} 1/2 \\ 1/3 \end{bmatrix}$$

for the lower vector on the right. Altogether, our system is

$$\begin{bmatrix} \boldsymbol{D} \\ \boldsymbol{Z}^T\boldsymbol{\Phi}^*\boldsymbol{\Phi} \end{bmatrix} \boldsymbol{x} = \begin{bmatrix} 0 & 0 & 2 \\ 1 & 1/2 & 1/3 \\ 1/2 & 1/3 & 1/4 \end{bmatrix} \boldsymbol{x} = \begin{bmatrix} 3 \\ 1/2 \\ 1/3 \end{bmatrix} = \begin{bmatrix} \boldsymbol{a} \\ \boldsymbol{Z}^T\boldsymbol{\Phi}^*\beta \end{bmatrix},$$

which has unique solution $\boldsymbol{x} = [1/4 \quad -1/2 \quad 3/2]^T$. Using the values from the solution vector as coefficients in (5.28), we obtain the approximate solution $\hat{u}(t) = (1/4) \cdot \phi_1 - (1/2) \cdot \phi_2 + (3/2) \cdot \phi_3 = 1/4 - t/2 + 3t^2/2 = u(t)$. The collocation method with the monomial basis functions gives the exact answer for this simple problem.

For both this method and the Galerkin method, the choice of basis functions for our solution approximation can affect the accuracy of the solution. For example, if we repeat the same example with basis functions 1, $\sin(t)$ and $\cos(t)$, our calculation proceeds as follows. First, our second derivatives are

$$\begin{aligned} \phi_1'' &= 0, \\ \phi_2'' &= -\sin(t), \\ \phi_3'' &= -\cos(t). \end{aligned}$$

Putting these equations into (5.30) above and evaluating the result at $t = 1/2$,

we have our initial system, to four digits of accuracy,

$$\boldsymbol{D}\boldsymbol{x} = \begin{bmatrix} 0 & -0.4794 & -0.8776 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = [3] = \boldsymbol{a}.$$

A null basis for this problem matrix is

$$\boldsymbol{Z} = \begin{bmatrix} 1 & 0 \\ 0 & 2.086 \\ 0 & -1.139 \end{bmatrix}.$$

Evaluating the integrals in (5.27), we obtain

$$\begin{aligned} \boldsymbol{Z}^T(\boldsymbol{\Phi}^*\boldsymbol{\Phi}) &= \begin{bmatrix} 1.000 & 0 & 0 \\ 0 & 2.086 & -1.139 \end{bmatrix} \begin{bmatrix} 1.000 & 0.4597 & 0.8415 \\ 0.4597 & 0.2727 & 0.3540 \\ 0.8415 & 0.3540 & 0.7273 \end{bmatrix} \\ &= \begin{bmatrix} 1 & 0.4597 & 0.8415 \\ 4.657 \times 10^{-4} & 0.1656 & -0.08995 \end{bmatrix} \end{aligned}$$

for the lower submatrix on the left and

$$\boldsymbol{Z}^T(\boldsymbol{\Phi}^*\beta) = \begin{bmatrix} 1.000 & 0 & 0 \\ 0 & 2.086 & -1.139 \end{bmatrix} \begin{bmatrix} 0.5000 \\ 0.3012 \\ 0.3818 \end{bmatrix} = \begin{bmatrix} 0.5000 \\ 0.1934 \end{bmatrix}$$

for the lower vector on the right. Altogether, our system is

$$\begin{aligned} \begin{bmatrix} \boldsymbol{D} \\ \boldsymbol{Z}^T\boldsymbol{\Phi}^*\boldsymbol{\Phi} \end{bmatrix} \boldsymbol{x} &= \begin{bmatrix} 0 & -0.4794 & -0.8776 \\ 1 & 0.4597 & 0.8415 \\ 4.657 \times 10^{-4} & 0.1656 & -0.08995 \end{bmatrix} \boldsymbol{x} \\ &= \begin{bmatrix} 3.0000 \\ 0.5000 \\ 0.1934 \end{bmatrix} = \begin{bmatrix} \boldsymbol{a} \\ \boldsymbol{Z}^T\boldsymbol{\Phi}^*\beta \end{bmatrix}, \end{aligned}$$

which has unique solution $\boldsymbol{x} = [2.650 \quad -0.1755 \quad -2.460]^T$. Using the values from the solution vector as coefficients in (5.28), we obtain the approximate solution $\hat{u}(t) = 2.650 - 0.1755\sin(t) - 2.460\cos(t)$. This approximation and the exact solution are graphed in Figure 5.5.

Next, we turn to the two-dimensional example given in Equation (5.20). We will use polynomial basis functions, specifically, the products of monomial basis functions in each variable. We describe convergence in terms of the number of basis functions in each dimension. For example, if the number of basis functions (nb) in a dimension is 2, the total number of basis functions is $2^2 = 4$, and the basis functions are $1 \cdot 1 = 1$, $s \cdot 1 = s$, $1 \cdot t = t$, and $s \cdot t = st$. Figure 5.6 plots the error in the collocation approximation for various values of nb. The error is calculated as the infinity norm of the absolute difference between the actual

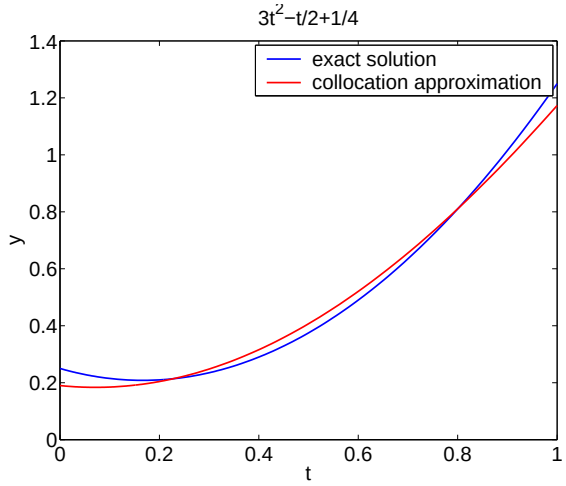

Figure 5.5: Exact and collocation method solutions to 1-D Poisson example

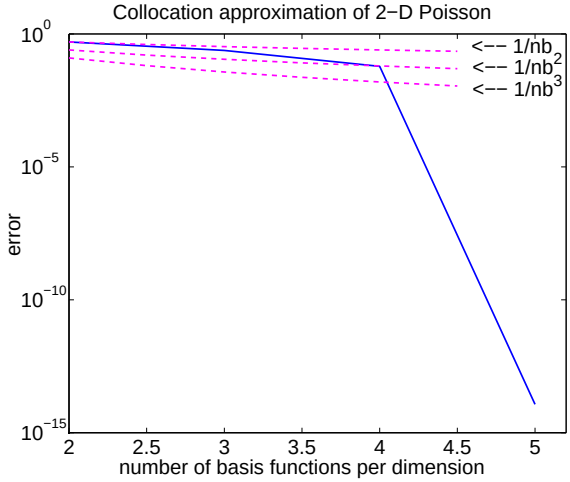

Figure 5.6: Error in collocation approximation for 2-D Poisson example

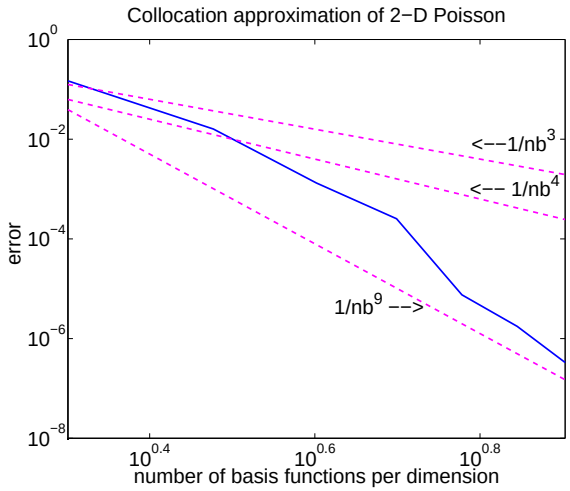

Figure 5.7: Error in collocation approximation for second 2-D Poisson example

and the approximate solution values at uniformly distributed mesh points over the domain, with 15 grid points in each dimension. When the function s^2t^4 is added to the set of basis functions, the solution can be represented exactly, and the error is essentially zero.

It is difficult to estimate convergence for the collocation method from the error graph given by the two-dimensional example given above. The following example uses the same differential operator but has a solution that cannot be exactly represented by polynomial basis functions. Let

$$\begin{aligned} \mathcal{D} &= \Delta = \tfrac{\partial^2}{\partial s^2} + \tfrac{\partial^2}{\partial t^2}, \\ \Omega &= (0,1) \times (0,1), \\ \alpha &= (4s^2 + 4t^2)\exp(s^2 - t^2), \\ \beta &= \exp(s^2 - t^2). \end{aligned} \tag{5.31}$$

The solution here is $\sin(s^2+t)$, and the error graph, with error calculated as we described above, is given in Figure 5.7. These approximations used uniformly-spaced points throughout the domain. The dashed lines represent convergence curves of $\mathcal{O}(h^3)$, $\mathcal{O}(h^4)$, and $\mathcal{O}(h^9)$, where $h = 1/nb$. This graph has a non-smooth convergence curve, but it clearly does better than quadratic convergence in the number of basis functions per dimension and seems to follow roughly the $\mathcal{O}(h^9)$ convergence curve. Collocation approximations may converge to the solution at different rates depending on the basis functions and on the spacing of the collocation points.

5.4 Discrete Null Bases

Along with discrete operators come discrete null bases. We now turn to the question of how the null spaces represented by discrete null bases derived from

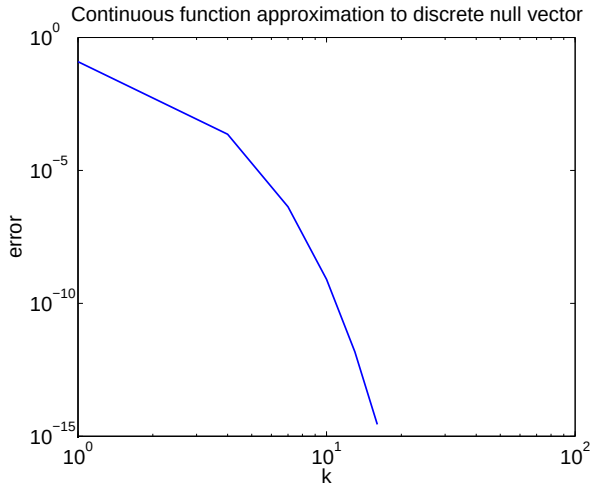

Figure 5.8: Difference between a specific null function and null vector

discretized differential operators relate to the null spaces of the corresponding continuous operators. Related work linking continuous and discrete variational forms is presented by Bochev and Lehoucq in [7]. We have not answered the question in full for null spaces, but we do have part of the answer for a Galerkin discretization of the Laplacian operator.

A harmonic function is by definition in the null space of the Laplacian operator, and any function of the form $\alpha \sin(\lambda s) \sinh(\lambda t)$ is a harmonic function for any real values of α and λ. One way to relate continuous and discrete null spaces is to show that continuous null functions have corresponding null vectors. We have found such a correspondence for a harmonic function of the form given above with $\alpha = \frac{2}{\sqrt{3}\sinh(\frac{2k\pi}{3}(n-1))}$ and $\lambda = \frac{2k\pi}{3}(n-1)$, where $k = 3i - 2$ for any positive integer i. A specific null vector, with zero values throughout the domain except for at the nodes on one edge, where the nodal values cycle through 0, 1 and -1, correlates closely with the values of this function,

$$\frac{2}{\sqrt{3}\sinh(\frac{2k\pi}{3}(n-1))} \sin(\frac{2k\pi}{3}(n-1)s) \cdot \sinh(\frac{2k\pi}{3}(n-1)t), \tag{5.32}$$

at the grid points of a uniform $n \times n$ discretization. And, as the value of k increases, the null vector and the discrete values of the null function converge, as shown in Figure 5.8 for the case $n = 5$. The error in the plot is calculated as the max norm of the difference between the null vector and the vector of discrete values of the continous null function at the grid points. A similar null vector approximates Function (5.32) when $k = 3i - 1$ for any positive integer i, and null vectors with the opposite orientation approximate another slightly varied version of the function.

Ideally, we would relate every null vector and null function to a corresponding element in the continous or discrete space, respectively. This would confirm

there are no spurious discrete null vectors and allow us calculate some measure of error between our discrete null space and the continous one. (A measure of error might be found using approximation theory [41], but we did not pursue that.) We have not been able to equate other specific null functions and null vectors (relating the continous to the discrete), but under a mild condition, we can relate discrete null vectors to continuous null functions (relating the discrete to the continous). Specifically, we prove that all the discrete null vectors of the matrix formed by a linear, two-dimensional, hat function discretization of the Laplacian on an uniformly-spaced $n \times n$ grid correspond to continuous null functions of the Laplacian operator. The following theorem shows that if the span of the basis functions used in the discrete approximation is a superset of the span of the Laplacian of those functions (a natural condition for many types of basis functions), then the linear combinations of null vectors and basis functions are null functions of the continuous operator.

Theorem 2 *For the continuous problem $\mathcal{D}\boldsymbol{\Phi}\boldsymbol{x} = 0$, let $\boldsymbol{\Phi} = [\,\phi_1 \quad \cdots \quad \phi_n\,]$ be a set of functions spanning the approximate solution space such that*

$$\text{span}(\Phi) \supseteq \text{span}(\mathcal{D}\Phi),$$

and let $\boldsymbol{D}$ be the matrix with entries $d_{ij} = \langle \mathcal{D}\phi_j, \phi_i \rangle$. Then $\boldsymbol{D}\boldsymbol{x} = \boldsymbol{0}$ implies $\mathcal{D}\boldsymbol{\Phi}\boldsymbol{x} = 0$.

Proof: Let $\boldsymbol{x}$ be any nonzero vector in the null space of $\boldsymbol{D}$,

$$\boldsymbol{D}\boldsymbol{x} = \boldsymbol{0}.$$

Then $\boldsymbol{x}$ is orthogonal to each row of $\boldsymbol{D}$:

$$x_1\langle \mathcal{D}\phi_1, \phi_i \rangle + x_2\langle \mathcal{D}\phi_2, \phi_i \rangle + \cdots + x_n\langle \mathcal{D}\phi_n, \phi_i \rangle = 0, \qquad i = 1, \ldots, n,$$

which by properties of inner products can be rewritten as

$$\langle x_1\mathcal{D}\phi_1 + x_2\mathcal{D}\phi_2 + \cdots + x_n\mathcal{D}\phi_n, \phi_i \rangle = 0, \qquad i = 1, \ldots, n.$$

Since $\mathcal{D}$ is a linear operator, this simplifies to

$$\langle \mathcal{D}\Phi\boldsymbol{x}, \phi_i \rangle = 0, \qquad i = 1, \ldots, n.$$

Assuming $\mathcal{D}\Phi\boldsymbol{x} \neq 0$, then

$$\text{span}([\mathcal{D}\Phi\boldsymbol{x}\ \Phi]) \supset \text{span}(\Phi),$$

by $\mathcal{D}\Phi\boldsymbol{x}$ orthogonal to ϕ_i for $i = 1, \ldots, n$. But, from the assumptions above,

$$\text{span}(\Phi) \supseteq \text{span}(\mathcal{D}\Phi),$$

implying

$$\mathcal{D}\Phi\boldsymbol{x} \in \text{span}(\Phi)$$

for any $\boldsymbol{x}$. This is a contradiction, so it follows that $\mathcal{D}\Phi\boldsymbol{x} = 0$. $\square$

6 Computing Null-Space Bases

Solving a discretized partial differential equation using the null-space method, as described in Chapter 4, requires a discrete null basis. There are two ways to approach this task. One approach is to find a null basis for the continuous operator and discretize it in the same way that we discretize the operator. A second approach is to find a basis for the null space of the discretized operator.

The first option may appear preferable at first. Finding a null basis for the continuous operator only once and then reusing it for several problems, discretizing the null basis in various ways according to the solution technique, seems preferable to finding a null basis specific to a particular discretization. In one dimension, finding a continuous null basis is often possible and simple. In those cases, a continuous null space may well be the better choice. For problems in more dimensions, however, differential operators often have infinite-dimensional null spaces. For such operators, finding a continuous null space to use for a discretized problem poses two difficulties. First, finding a basis of an infinite-dimensional space is nontrivial, though it is possible in some cases, as [21] demonstrates. Second, once a null basis is found, we face the task of choosing which null space functions to discretize. The discrete approximation must have a finite-dimensional null space by construction as a discrete problem, and as it is impossible to represent the entire infinite-dimensional space with a finite set of functions, we must select a subset of these continuous functions to discretize. The best method for choosing that subset is unclear, and additionally, in exploring this option we found that some linearly independent continuous functions may be nearly linearly dependent when discretized, which adds to the complexity of discretizing the infinite-dimensional space.

The second option, finding a null basis for the discretized operator, is the approach we use in our work. We not only wish to find a null basis, we also prefer a basis that has good numerical and computational properties. Specifically, we hope to find a sparse, well-conditioned null basis. In this chapter we first discuss in Section 6.1 how to find null bases of general matrices. Then in Section 6.2 we explain four important special cases for which we have found explicit null bases. Section 6.3 gives results related to computing and using null bases, and in Section 6.4 we consider issues concerning the conditioning of these null bases.

6.1 General Case

The task of finding the null basis of a matrix occurs in many contexts, and several options are available to us. For example, we could compute the singular value decomposition (SVD) of a matrix and take the right singular vectors corresponding to zero singular values as our null basis. In the case of an $m \times p$ underdetermined matrix, only m singular values exist and the $p - m$ singular vectors not corresponding to any singular values are used for the null basis in addition to any singular vectors corresponding to zero singular values. (This decomposition is the method used by the MATLAB function `null()`.) We could also generate a full QR factorization of the transpose of our matrix and take the right-most vectors of the matrix $\boldsymbol{Q}$ as our null basis. Both of these options provide a well-conditioned null basis, but the result is usually dense. A dense basis hinders our goals of keeping the solution time and storage requirements small, so these two methods are useful for small problems and for comparative purposes but are not good candidates for solving large systems unless computational resources are not a limiting factor.

Much attention has been given to the problem of finding a sparse null basis, and as it has been shown to be NP-hard to find the sparsest null basis [14], many heuristics have been developed to find "suitably" sparse null bases. The turnback algorithm for finding sparse null bases developed over several years ([30, 34, 52]), culminating in a paper by Berry, Heath, Kaneko, Lawo, Plemmons, and Ward in 1985 [6]. The turnback algorithm begins by passing through the matrix to find columns that are linearly dependent on previous columns using a QR factorization, without saving the matrix $\boldsymbol{Q}$. Next, it searches for a dependent set of columns, starting with a column marked in the first pass and adding previous columns one at a time, expanding the search set backward toward the first column until a dependency is found between columns in the search set. A null vector is determined from the dependent set using a small QR factorization updated and expanded when a vector is added to the search space. The method capitalizes on the idea that dependencies are often local within a matrix. The algorithm ensures that the set of null vectors is independent by constraining which columns may be added to the search for a dependent set; if a vector $\boldsymbol{a}$ was the last one added to a dependent set, it may not be used again. This does not create problems with the algorithm finding a complete set of null vectors because the vectors with which $\boldsymbol{a}$ formed a dependent set span $\boldsymbol{a}$.

In 1987, Gilbert and Heath [24] proposed and compared variations on the turnback method, offering options for preordering the matrix columns and providing better heuristics for finding dependent sets with few columns, leading to sparser null bases. The new heuristics do not search for dependencies using column order only, but they also consider the number of nonzero rows, and in some cases combinatorial matching algorithms play a role. Their results have been

the basis for much of our work in finding null bases. In the same year, another algorithm based on matching was published by Coleman and Pothen [15]. Coleman and Pothen focus on matching algorithms and the structure of the matrix, assuming no numeric cancellation. Their results closely paralleled the results of an algorithm in Gilbert and Heath's paper. Subsequently, Pothen [43] developed a faster method that produces sparser bases, but it is specific to the problem of finding null bases of equilibrium matrices. Plemmons and White [42] addressed parallel algorithms for the same problem in 1990. In 1993, Stern and Vavasis [49] proposed a method for finding sparse null bases using nested dissection.

Of the three approaches listed above, turnback, graph matching algorithms, and nested dissection, we have worked with the turnback method and its variations, primarily because an implementation was available to us. The code was written for research purposes, not as production code, but it gives useful comparisons for smaller problems, and it gave a good starting point for finding explicit bases. Presumably any of the methods listed above could provide a useful sparse null basis even for larger problems, given a proper implementation.

While the turnback method and its variations are effective for general matrices, we believe we can improve on the algorithm for some specific problems. In particular, in some problems we can exploit symmetry to derive two, four, or even eight null vectors from a single null vector without additional calculation. Similarly, Ye and Hall [59] manipulated the basic turnback to create a more symmetric basis. In practice, this idea enables us to find sparser null bases than the turnback algorithm finds. Since this improvement is specific to certain discretizations and problem domains, we have not automated the process, but its effects are evident in the explicit bases we have found for various operators. (See Section 6.2.)

Other work relevant to sparse null bases includes papers by Hoffman and McCormick [32], Adler et al. [3], McCormick [37], Chang and McCormick [11, 12], and Gondzio [25] which propose methods for making a sparse matrix sparser and a paper by Brualdi, Friedland, and Pothen [9] presenting a polynomial time algorithm to test if a set of elementary vectors is a null basis. The more recent work of Chang and McCormick is based on matching algorithms and looks for linear combinations of vectors that are sparser than the original vectors, resulting in a matrix with the same span as the original matrix but with fewer nonzero entries. Experimentation with this method has shown their algorithm to be ineffective for our problems, though the paper shows the method to be useful for many linear programming problems.

6.2 Explicit Bases

In general, we will use one of the methods explained above to find a null basis for the discretized differential equation we wish to solve. In four important special cases, however, we will exhibit an explicit null basis for the discrete operator. In

each of these cases, we need not apply a general method for finding a null basis as described above. Rather, we can form the null basis without even forming the problem matrix. In this section we explain the discretization of the operator and formation of explicit null vectors in each special case. In three cases, we additionally prove that the null vectors we describe form a null basis for the specific discrete operator.

6.2.1 Finite Difference 2-D Laplacian

First, we consider solving Poisson's equation in two dimensions. Our domain is the unit square, and we discretize the Laplacian operator with centered finite differences, using a five-point stencil and uniform grid spacing in both dimensions. As boundary conditions are unspecified, we use only stencils that are completely contained within our grid and discard any others. This method produces an underdetermined, rectangular matrix. We give formulas for explicitly producing a null basis for any matrix so formed from an $n \times n$ discretization and prove that it is indeed a null basis. We assume throughout that $n \geq 3$.

First, we present the formulas for null vectors.

Entries of our null vectors will be taken from the three-sum triangle. The three-sum triangle, $\boldsymbol{T}$, is a lower triangular matrix such that each entry on or below the diagonal is found by summing the values to the immediate west, north, and northwest of it. The table initially has 1's in the leftmost column. For each entry in a given row i and column j,

$$\boldsymbol{T}(i,j) = \begin{cases} 0 & i < j \\ 1 & j = 1 \\ \boldsymbol{T}(i-1,j) + \boldsymbol{T}(i-1,j-1) + \boldsymbol{T}(i,j-1) & \text{otherwise} \end{cases} . \tag{6.1}$$

The diagonal entries of this matrix are also known as the large Schroeder numbers [40, 48]. A portion of the three-sum triangle is shown in (6.2) below, where blank entries represent 0 values.

$$\begin{bmatrix} 1 & & & & & & \\ 1 & 2 & & & & & \\ 1 & 4 & 6 & & & & \\ 1 & 6 & 16 & 22 & & & \\ 1 & 8 & 30 & 68 & 90 & & \\ 1 & 10 & 48 & 146 & 304 & 394 & \\ 1 & 12 & 70 & 264 & 714 & 1412 & 1806 \end{bmatrix} \tag{6.2}$$

For purposes of the following proof, it is useful to think of null vectors in terms of their numerical values on the finite difference grid. We refer to the null vector grids as *null grids*, and we present the explicit formula in terms of values on the grid.

We have four types of null grids, $\boldsymbol{u}_k^n$, $\boldsymbol{v}_k^n$, $\boldsymbol{w}_k^n$, and $\boldsymbol{z}_k^n$, where n is the size of

the problem (an $n \times n$ discretization) and k specifies a particular null grid of the set. The null grids are described in terms of values on the finite-difference grid, so they are additionally indexed by rows and columns. For example, $\boldsymbol{u}_k^n(i,j)$ (for $1 \le i \le n$ and $1 \le j \le n$) corresponds to the value of $\boldsymbol{u}_k^n$ at the point $((i-1)h,(j-1)h)$, where $h = 1/(n-1)$.

For arbitrary n, then, we form null grids using $1 \le k < n$ in each of these formulas:

$$\boldsymbol{u}_k^n(i,j) = \begin{cases} 1, & i = k,\ j = 1 \\ (-1)^{j-1}\boldsymbol{T}(k+j-i-1, k-i-j+2), & \\ & 1 \le i \le k-1,\ 2 \le j \le k-i+1 \\ 0, & \text{elsewhere} \end{cases} \tag{6.3}$$

$$\boldsymbol{v}_k^n(i,j) = \begin{cases} 1, & i = 1,\ j = n-k+1 \\ (-1)^{i-1}\boldsymbol{T}(k-n+j+i-2, k-n-i+j+1), & \\ & 2 \le i \le j+k-n,\ n-k+2 \le j \le n \\ 0, & \text{elsewhere} \end{cases} \tag{6.4}$$

$$\boldsymbol{w}_k^n(i,j) = \begin{cases} 1, & i = n,\ j = k \\ (-1)^{n-i}\boldsymbol{T}(k-i-j+n, i-j+k-n+1), & \\ & n-k+j \le i \le n-1,\ 1 \le j \le k-1 \\ 0, & \text{elsewhere} \end{cases} \tag{6.5}$$

$$\boldsymbol{z}_k^n(i,j) = \begin{cases} 1, & i = n-k+1,\ j = n \\ (-1)^{n-j}\boldsymbol{T}(k+i-j-1, k+i+j-2n), & \\ & n-k+2 \le i \le n,\ 2n-k-i+1 \le j \le n-1 \\ 0, & \text{elsewhere} \end{cases} \tag{6.6}$$

Formulas (6.3)-(6.6) hide the simplicity of the null grids and their relationships to each other. The four formulas represent grids that are exactly the same except for the corner from which they originate, i.e., they are rotationally symmetric. For null grids originating from the same corner, when k is increased by 1, the new null grid is just a shift of the nonzero values of the previous null grid with one row of nonzeros added, filled in with the numbers from the appropriate anti-diagonal of (6.2). The skew diagonals used for the null grids (every other one) are highlighted in red in (6.2).

For example, Figure 6.1 displays the null grids for $n = 5$, in order from left to right, $\boldsymbol{u}$, $\boldsymbol{v}$, $\boldsymbol{w}$ $\boldsymbol{z}$, and top to bottom, $k = 1, 2, 3, 4$.

Now that we have presented the null grids, we prove in a series of theorems and lemmas that they are indeed null vectors and that they form all or most of a null basis. First, we note that the finite difference stencils for an $n \times n$ grid are of the form

$$(n-1)^2 \begin{bmatrix} \ddots & \vdots & \vdots & \vdots & \ddots \\ \cdots & 0 & 1 & 0 & \cdots \\ \cdots & 1 & -4 & 1 & \cdots \\ \cdots & 0 & 1 & 0 & \cdots \\ \ddots & \vdots & \vdots & \vdots & \ddots \end{bmatrix},$$

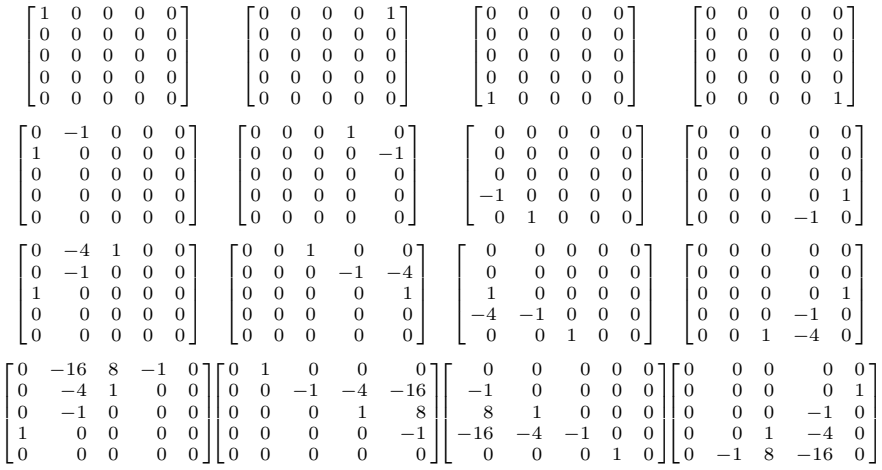

Figure 6.1: Null grids for $n = 5$.

where all entries represented by dots are zero. In order to have a null grid g, then, we need

$$(n-1)^2(1 \cdot g(i-1,j) + 1 \cdot g(i,j-1) - 4 \cdot g(i,j) + 1 \cdot g(i,j+1) + 1 \cdot (i+1,j)) = 0$$

for all $2 \leq i, j \leq n-1$. It is clear that the constant $(n-1)^2$ can be removed from the equation above and we omit it in our proofs. We refer to each equation formed from a finite difference stencil as a *constraint* and the part remaining on the left-hand side after we remove the scalar as a *weighted sum*.

Theorem 3 *$\boldsymbol{u}_k^n$ is a null grid for the specified problem of size n for all $n \geq 3$ and $1 \leq k < n$.*

Proof: The proof proceeds by induction on n.

Base cases: $\boldsymbol{u}_1^3, \boldsymbol{u}_2^3$. For the problem with $n = 3$, we have one finite difference stencil,

$$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}. \tag{6.7}$$

In order to have a null grid, we must satisfy the constraint $\boldsymbol{u}(2,1) + \boldsymbol{u}(1,2) + -4\boldsymbol{u}(2,2) + \boldsymbol{u}(3,2) + \boldsymbol{u}(2,3) = 0$.

The base case grids, $\boldsymbol{u}_1^3$ and $\boldsymbol{u}_2^3$, are shown below with the pertinent values highlighted in blue.

$$u_1^3 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \qquad u_2^3 = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

For $\boldsymbol{u}_1^3$, we have $0 + 0 - 4 \cdot 0 + 0 + 0 = 0$, so the condition is satisfied. For $\boldsymbol{u}_2^3$, we have $1 - 1 - 4 \cdot 0 + 0 + 0 = 0$. Again, we have a null grid.

Induction hypothesis: Assume $\boldsymbol{u}_k^n$ is a null grid for $1 \leq k < n$. We will show that $\boldsymbol{u}_k^{n+1}$ is a null grid for $1 \leq k < n+1$.
Step 1: First we show $\boldsymbol{u}_k^{n+1}$ is a null grid for $1 \leq k < n$. By the induction hypothesis, we know that $\boldsymbol{u}_k^n$ is a null grid and thus

$$\boldsymbol{u}_k^n(i,j-1) + \boldsymbol{u}_k^n(i-1,j) - 4\boldsymbol{u}_k^n(i,j) + \boldsymbol{u}_k^n(i+1,j) + \boldsymbol{u}_k^n(i,j+1) = 0$$

for $2 \leq i,j \leq n-1$. By inspection, since n does not appear in formula (6.3), $\boldsymbol{u}_k^{n+1}$ has exactly the same nonzero positions and values as $\boldsymbol{u}_k^n$ and

$$\boldsymbol{u}_k^{n+1}(i,j-1) + \boldsymbol{u}_k^{n+1}(i-1,j) - 4\boldsymbol{u}_k^{n+1}(i,j) + \boldsymbol{u}_k^{n+1}(i+1,j) + \boldsymbol{u}_k^{n+1}(i,j+1) = 0$$

for $2 \leq i \leq n-1$ and $2 \leq j \leq n-1$. Thus, $\boldsymbol{u}_k^{n+1}$ satisfies all the constraints that are duplicates of those on the $n \times n$ grid.

Since we are now working on the $(n+1)\times(n+1)$ grid, we must also show that constraints associated with the finite difference stencils with centers along the gridlines with $i = n$ and $j = n$ are satisfied. As we now show, these conditions are all met because the grid values associated with the stencils are all 0.

First, suppose we have a stencil centered at (n,j) with $2 \leq j \leq n$. For $k < n$, from the limits of the variables for (6.3), we see that the maximum row with a nonzero value is row $n-2$ except when $j = 1$. Thus $\boldsymbol{u}_k^{n+1}(n,j) = 0$ and $\boldsymbol{u}_k^{n+1}(n+1,j) = 0$ for all j, and $\boldsymbol{u}_k^{n+1}(n-1,j) = 0$ for $j \geq 2$, which implies

$$\boldsymbol{u}_k^{n+1}(n,j-1) + \boldsymbol{u}_k^{n+1}(n-1,j) - 4\boldsymbol{u}_k^{n+1}(n,j) + \boldsymbol{u}_k^{n+1}(n+1,j) + \boldsymbol{u}_k^{n+1}(n,j+1) = 0$$

for $2 \leq j \leq n$, satisfying the constraints of all the finite difference stencils centered at $i = n$.
Similarly, the maximum column of $\boldsymbol{u}_k^{n+1}(i,n)$ with a nonzero value is $n-1$ and is obtained only when $i = 1$. In general the maximum column with a nonzero value is $n-2$. Thus $\boldsymbol{u}_k^{n+1}(i,n) = 0$ and $\boldsymbol{u}_k^{n+1}(i,n+1) = 0$ for all i, and $\boldsymbol{u}_k^{n+1}(i,n+1) = 0$ for $i \geq 2$. It follows that

$$\boldsymbol{u}_k^{n+1}(i,n-1) + \boldsymbol{u}_k^{n+1}(i-1,n) - 4\boldsymbol{u}_k^{n+1}(i,n) + \boldsymbol{u}_k^{n+1}(i+1,n) + \boldsymbol{u}_k^{n+1}(i,n+1) = 0$$

for $2 \leq i \leq n$, satisfying the constraints centered at $j = n$. Putting these pieces together, then, all of the null constraints are satisfied, and $\boldsymbol{u}_k^{n+1}$ is confirmed to be a null grid for $1 \leq k < n$.

Step 2: To complete the proof, we must show that $\boldsymbol{u}_n^{n+1}$ is a null grid. From (6.3) we see that $\boldsymbol{u}_n^{n+1}(i,j) = \boldsymbol{u}_{n-1}^{n+1}(i-1,j)$ for $1 \leq j \leq n+1$ and $2 \leq i \leq n+1$. That is, most of $\boldsymbol{u}_n^{n+1}$ is a shift of $\boldsymbol{u}_{n-1}^{n+1}$. As the finite difference stencils are the same across the domain, and since we know from Step 1 that $\boldsymbol{u}_{n-1}^{n+1}$ is a null grid, $\boldsymbol{u}_n^{n+1}$ meets the requirements for a null grid for all of the finite difference stencils centered from $3 \leq i \leq n$ and $2 \leq j \leq n$. We need to confirm, then, only that the stencils centered along the row $i = 2$ also have a weighted sum of zero.

The weighted sum looks like this (where $2 \leq j \leq n$):

$$S = \boldsymbol{u}_n^{n+1}(1,j) + \boldsymbol{u}_n^{n+1}(2,j-1) - 4\boldsymbol{u}_n^{n+1}(2,j) + \boldsymbol{u}_n^{n+1}(2,j+1) + \boldsymbol{u}_n^{n+1}(3,j).$$

Substituting according to (6.3) we find

$$\begin{aligned} S = (-1)^j(&\boldsymbol{T}(n+j-2,n-j+1) - \boldsymbol{T}(n+j-4,n-j+1) \\ &-4\boldsymbol{T}(n+j-3,n-j) - \boldsymbol{T}(n+j-2,n-j-1) + \boldsymbol{T}(n+j-4,n-j-1)). \end{aligned}$$

Now suppose for ease of notation that

$$\begin{aligned} \boldsymbol{T}(n+j-4,n-j-1) &= a, \\ \boldsymbol{T}(n+j-4,n-j) &= b, \\ \boldsymbol{T}(n+j-4,n-j+1) &= c, \\ \boldsymbol{T}(n+j-3,n-j-1) &= d, \\ \boldsymbol{T}(n+j-2,n-j-1) &= f, \end{aligned}$$

for some values a, b, c, d, and f. We do not want to define values outside of the table so we limit $n+j-4 \geq 1$ and $n-j-1 \geq 1$, which implies $2 \leq j \leq n-2$. We will address the outliers later. By the construction of $\boldsymbol{T}$,

$$\begin{aligned} \boldsymbol{T}(n+j-3,n-j) &= a+b+d, \\ \boldsymbol{T}(n+j-3,n-j+1) &= a+2b+c+d, \\ \boldsymbol{T}(n+j-2,n-j) &= a+b+2d+f, \\ \boldsymbol{T}(n+j-2,n-j+1) &= 3a+4b+c+4d+f. \end{aligned}$$

Here again, we must check validity of indices. The reciprocally-defined portion of $\boldsymbol{T}$ (which we use in the derivation above) requires the first index to be greater than or equal to the second. Checking indices, we find that $j \geq 2$ guarantees that this requirement is met in all four cases. Writing the above, which is an arbitrary 3×3 block of $\boldsymbol{T}$, in matrix form, we have

$$\begin{bmatrix} a & b & c \\ d & a+b+d & a+2b+c+d \\ f & a+b+2d+f & 3a+4b+c+4d+f \end{bmatrix}.$$

Substituting the pertinent values into the equation for S above,

$$\begin{aligned} S &= (-1)^j(3a+4b+c+4d+f-c-4\cdot(a+b+d)-f+a), \\ S &= (-1)^j(3a-4a+a+4b-4b+c-c+4d-4d+f-f), \\ S &= 0. \end{aligned}$$

In general, then, we have shown that the stencils along the grid line $i = 2$

have a weighted sum of zero, but a few remaining stencils at the right and left of the grid need special attention. For the finite difference stencil centered at $(2,2)$, $\boldsymbol{u}_n^{n+1}(2,1)$ is defined to be zero; it is not given in terms of $\boldsymbol{T}$. In this case, however, the pertinent value of $\boldsymbol{T}$ is also zero, so the weighted sum for the stencil is equal to that derived above, namely 0. Now we address the cases left out above by the limits on j. For the null grid centered at $(2,n)$, we have nonzeros at $(1,n)$ and $(2,n-1)$. The weighted sum becomes

$$(-1)^{n+1}(\boldsymbol{T}(2n-2,1)-\boldsymbol{T}(2n-4,1))=(-1)^{n+1}(1-1)=0.$$

For the null grid centered at $(2,n-1)$, we have nonzeros at $(1,n-1),(2,n-2)$, and $(2,n-1)$. Here the weighted sum becomes

$$(-1)^{n+1}(-\boldsymbol{T}(2n-3,2)+4\boldsymbol{T}(2n-4,1)+\boldsymbol{T}(2n-5,2)). \tag{6.8}$$

Using a method similar to that above, assuming $\boldsymbol{T}(2n-5,2)=c$ and knowing that $\boldsymbol{T}(i,1)=1$, expression (6.8) simplifies to

$$(-1)^{n+1}(-(c+4)+4\cdot 1+c)=(-1)^{n+1}(0)=0.$$

Thus, all the constraints are met and $\boldsymbol{u}_n^{n+1}$ is confirmed to be a null grid. This, together with Step 1, proves that given our induction hypothesis, $\boldsymbol{u}_k^{n+1}$ is a null grid for $1\le k<n+1$. By induction, then, $\boldsymbol{u}_k^n$ is a null grid for all $n\ge 3$ and $1\le k<n$. □

Theorem 4 *$\boldsymbol{v}_k^n$, $\boldsymbol{w}_k^n$, $\boldsymbol{z}_k^n$ are null grids for the specified problem of size n for all $n\ge 3$ and $1\le k<n$.*

Proof: True by rotational symmetry of the null grids and the constraints. □

Lemma 4 *The grid of n^2 1's is a null grid for the specified problem of size n for $n\ge 3$.*

Proof: For each finite difference stencil, the weighted sum will be

$$1\cdot 1+1\cdot 1-4\cdot 1+1\cdot 1+1\cdot 1=0. \qquad \square$$

At this point, we transition from thinking about null grids to null vectors. While we still refer to $\boldsymbol{u}_k^n$, $\boldsymbol{v}_k^n$, $\boldsymbol{w}_k^n$, and $\boldsymbol{z}_k^n$, we now consider them as vectors that form columns of a matrix. We show that the matrix formed by concatenating null vectors is a basis for the null space. The null vectors are added to the matrix in the order that they are shown in the example following formulas (6.3)-(6.6), that is, $\boldsymbol{u}_1^n$, $\boldsymbol{v}_1^n$, $\boldsymbol{w}_1^n$, $\boldsymbol{z}_1^n$, $\boldsymbol{u}_2^n$, $\boldsymbol{v}_2^n$, ..., and they are generally added in groups of four corresponding to k values. The order of the columns is important to the proof because we show that in this formation there is an embedded upper triangular

matrix in the larger matrix. The rows that form the embedded triangular matrix are identified with respect to the order of the columns.

First, we prove that the matrix of the first $4(n-2)$ vectors contains an embedded upper triangular matrix with nonzero diagonal entries.

Lemma 5 *For all $n \geq 3$, $\boldsymbol{B} = [\boldsymbol{u}_1^n \ \boldsymbol{v}_1^n \ \boldsymbol{w}_1^n \ \boldsymbol{z}_1^n \ \cdots \ \boldsymbol{u}_{n-2}^n \ \boldsymbol{v}_{n-2}^n \ \boldsymbol{w}_{n-2}^n \ \boldsymbol{z}_{n-2}^n]$ contains an embedded upper triangular matrix with nonzero diagonal entries.*

Proof (by induction): Base case: $n = 3$. Supposing nodes are numbered left to right, top to bottom,

$$\boldsymbol{B} = [\boldsymbol{u}_1^3 \ \boldsymbol{v}_1^3 \ \boldsymbol{w}_1^3 \ \boldsymbol{z}_1^3] = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

By inspection, $\boldsymbol{B}$ has an imbedded identity matrix, which is upper triangular and has nonzero diagonal entries. The nonzero values are in the rows corresponding to the grid points $(1,1), (1,n), (n,1)$, and (n,n).

Induction hypothesis: Assume

$$\boldsymbol{B} = [\boldsymbol{u}_1^n \ \boldsymbol{v}_1^n \ \boldsymbol{w}_1^n \ \boldsymbol{z}_1^n \ \cdots \ \boldsymbol{u}_{n-2}^n \ \boldsymbol{v}_{n-2}^n \ \boldsymbol{w}_{n-2}^n \ \boldsymbol{z}_{n-2}^n]$$

has an embedded upper triangular matrix with nonzero diagonal entries. Suppose the nodes are numbered according to some mapping f such that $p_{i,j} = f(i,j)$, where i and j are grid indices and $p_{i,j}$ is the row (i.e., vector index) corresponding to the (i,j)th grid location. Also, let

$$\hat{\boldsymbol{B}} = [\boldsymbol{B} \ \boldsymbol{u}_{n-1}^{n+1} \boldsymbol{v}_{n-1}^{n+1} \ \boldsymbol{w}_{n-1}^{n+1} \ \boldsymbol{z}_{n-1}^{n+1}],$$

that is,

$$\hat{\boldsymbol{B}} = [\boldsymbol{u}_1^{n+1} \ \boldsymbol{v}_1^{n+1} \ \boldsymbol{w}_1^{n+1} \ \boldsymbol{z}_1^{n+1} \ \cdots \ \boldsymbol{u}_{n-2}^{n+1} \ \boldsymbol{v}_{n-2}^{n+1} \ \boldsymbol{w}_{n-2}^{n+1} \ \boldsymbol{z}_{n-2}^{n+1} \ \boldsymbol{u}_{n-1}^{n+1} \boldsymbol{v}_{n-1}^{n+1} \ \boldsymbol{w}_{n-1}^{n+1} \ \boldsymbol{z}_{n-1}^{n+1}].$$

Explanation: We will show that a certain set of rows of $\hat{\boldsymbol{B}}$ forms an embedded upper triangular matrix. Each node in the finite difference grid corresponds to a row in $\hat{\boldsymbol{B}}$, and each null grid/null vector corresponds to a column in $\hat{\boldsymbol{B}}$. We already know the order of the columns. For each column, then, we will select a row $p_{i,j}$ that is nonzero in the (i,j)th grid location for the associated column and zero in the (i,j)th grid location for all previous columns. While the proof follows by induction, the final order of the rows chosen follows the pattern shown

below for $n = 6$. The nodes correspond to rows and the numbers correspond to the order in which the rows are selected for the embedded matrix. The corner nodes are used for the first four rows (starting from the top left-hand corner and circling counter-clockwise), and we spiral inward from there.

$$\begin{bmatrix}
1 & \rightarrow & 5 & & \cdot & \cdot & & 8 & \leftarrow & 4 \\
 & & \downarrow & & & & & \downarrow & & \\
\cdot & & 9 & \rightarrow & 13 & 16 & \leftarrow & 12 & & \cdot \\
 & & & & \downarrow & \downarrow & & & & \\
\cdot & & \cdot & & 17 & 20 & & \cdot & & \cdot \\
 & & & & & & & & & \\
\cdot & & \cdot & & 18 & 19 & & \cdot & & \cdot \\
 & & & & \uparrow & \uparrow & & & & \\
\cdot & & 10 & \rightarrow & 14 & 15 & \leftarrow & 11 & & \cdot \\
 & & \uparrow & & & & & \uparrow & & \\
2 & \rightarrow & 6 & & \cdot & \cdot & & 7 & \leftarrow & 3
\end{bmatrix}.$$

Claim 1: First, we will show that the embedded matrix remains intact (i.e., the same rows can be used) from $\boldsymbol{B}$ to the first $4(n-2)$ columns of $\hat{\boldsymbol{B}}$, even though we are increasing to a grid of size $n+1$. Later we will consider the four additional null grids of $\hat{\boldsymbol{B}}$, $\boldsymbol{u}^n_{n-1}, \boldsymbol{v}^n_{n-1}, \boldsymbol{w}^n_{n-1}, \boldsymbol{z}^n_{n-1}$.

Without loss of generality, we consider the column that is the null vector $\boldsymbol{u}^n_m$. We know by the induction hypothesis that there is a row of $\boldsymbol{B}$, $p_{a,b}$ for some values of a and b, such that $\boldsymbol{u}^n_m(a,b) \neq 0$ and vectors to the left of $\boldsymbol{u}^n_m$ are zero at the gridpoint (a,b). That means $\boldsymbol{u}^n_k(a,b) = \boldsymbol{v}^n_k(a,b) = \boldsymbol{w}^n_k(a,b) = \boldsymbol{z}^n_k(a,b) = 0$ for $k < m$. We now show that the same holds for $\hat{\boldsymbol{B}}$ and $\boldsymbol{u}^{n+1}_m$.

First, we consider the $\boldsymbol{u}^{n+1}_k$ grids. All of the $\boldsymbol{u}^{n+1}_k$ grids have nonzeros in exactly the same locations as the $\boldsymbol{u}^n_k$ grids, so by construction and the hypotheses above, $\boldsymbol{u}^{n+1}_m(a,b) \neq 0$ and $\boldsymbol{u}^{n+1}_k(a,b) = 0$ for $k < m$.

Now we consider the $\boldsymbol{w}^{n+1}_k$ null grids. It is clear from (6.5) that $\boldsymbol{w}^{n+1}_k(i,j) = \boldsymbol{w}^n_k(i-1,j)$. In other words,

$$\boldsymbol{w}^n_k(i-1,j) = 0 \Rightarrow \boldsymbol{w}^{n+1}_k(i,j) = 0.$$

We also know from (6.3) and constraints on (i,j) (a position where $\boldsymbol{u}^n_k$ is nonzero; $i < n$) that

$$\boldsymbol{w}^n_k(i,j) = 0 \Rightarrow \boldsymbol{w}^n_k(i-1,j) = 0.$$

Thus,

$$\boldsymbol{w}^n_k(a,b) = 0 \Rightarrow \boldsymbol{w}^{n+1}_k(a,b) = 0,$$

and we will not introduce any new nonzero positions into row $p_{a,b}$ when we shift to the $(n+1) \times (n+1)$ grid.

Similarly,

$$\boldsymbol{v}^{n+1}_k(i,j) = \boldsymbol{v}^n_k(i,j-1),$$

and for $i > 1$,

$$\boldsymbol{v}_k^n(i,j) = 0 \Rightarrow \boldsymbol{v}_k^n(i,j-1) = 0.$$

Thus for $a > 1$,

$$\boldsymbol{v}_k^n(a,b) = 0 \Rightarrow \boldsymbol{v}_k^{n+1}(a,b) = 0.$$

The premise is true by the hypotheses above, so for $a > 1$, $\boldsymbol{v}_k^{n+1}(a,b) = 0$. For the case where $a = 1$,

$$\boldsymbol{v}_{k-1}^n(1,b) = 0 \Rightarrow \boldsymbol{v}_k^{n+1}(1,b) = 0.$$

We know that $\boldsymbol{v}_{k-1}^n(a,b)$ is a previous value in this row, so its value must be zero, implying $\boldsymbol{v}_k^{n+1}(a,b) = 0$.

The nonzero entries in the $\boldsymbol{z}_k^{n+1}$ vectors do not overlap with those in the $\boldsymbol{u}_k^{n+1}$ vectors. Any row that is nonzero for some grid location in $\boldsymbol{u}_k^{n+1}$ will be zero in all of the $\boldsymbol{z}_k^{n+1}$ columns.

Thus we conclude that row $p_{a,b}$ of $\hat{\boldsymbol{B}}$ is zero to the left of column $\boldsymbol{u}_m^n$. This argument, while the letters will be different depending on the first column chosen, holds for all columns of $\boldsymbol{B}$. This completes the proof of Claim 1.

Claim 2: Now we consider the four right-most vectors of $\hat{\boldsymbol{B}}$ (those associated with $k = n-1$) and show that rows can be selected to complete the embedded upper triangular matrix with nonzero diagonal entries.

Case 1: Assume n is even. In this case, it can be shown that the rows $f(\frac{n}{2},\frac{n}{2}), f(\frac{n}{2},\frac{n}{2}+2), f(\frac{n}{2}+2,\frac{n}{2}), f(\frac{n}{2}+2,\frac{n}{2}+2)$, associated with the columns $\boldsymbol{u}_{n-1}^{n+1}$, $\boldsymbol{v}_{n-1}^{n+1}$, $\boldsymbol{w}_{n-1}^{n+1}$, and $\boldsymbol{z}_{n-1}^{n+1}$, respectively, are nonzero in their associated columns and zero for all previous columns. We will demonstrate this for column $\boldsymbol{u}_{n-1}^{n+1}$ and row $p_{\frac{n}{2},\frac{n}{2}}$. The same can be shown for the other rows using similar arguments.

First, we show $\boldsymbol{u}_{n-1}^{n+1}$ is nonzero in the given row. From (6.3) we know that

$$\boldsymbol{u}_{n-1}^{n+1}(\frac{n}{2},\frac{n}{2}) = \pm\boldsymbol{T}(n-2,1) = \pm 1 \neq 0.$$

Now we must show that all previous vectors are zero in this row. We do this based on the variable limits for nonzeros in formulas (6.3)-(6.6), the values of i and j (both equal to $n/2$ here), and the knowledge that vectors to the left of this one in $\hat{\boldsymbol{B}}$ are formed with $k < n-1$. By showing that the limits do not allow nonzero values in the given row for these vectors, we prove that the values must be zero.

First, we consider $\boldsymbol{u}_k^{n+1}$. For a nonzero value with $i = \frac{n}{2}$, we would need

$$\begin{aligned}
j &\leq k - \frac{n}{2} + 1, \\
\frac{n}{2} &\leq k - \frac{n}{2} + 1, \quad \text{(substituting in the value for } j\text{)} \\
n &\leq k + 1, \\
n-1 &\leq k.
\end{aligned}$$

For a nonzero value in $\boldsymbol{v}_k^{n+1}$ with $j = \frac{n}{2}$, we would need

$$\begin{aligned} i &\leq \frac{n}{2} + k - (n+1), \\ \frac{n}{2} &\leq \frac{n}{2} + k - n - 1, \\ n+1 &\leq k. \end{aligned}$$

For $\boldsymbol{w}_k^{n+1}$ with $j = \frac{n}{2}$, we would need

$$\begin{aligned} i &\geq n + 1 - k + \frac{n}{2}, \\ \frac{n}{2} &\geq \frac{3n}{2} - k + 1, \\ -n &\geq -k + 1, \\ n+1 &\leq k. \end{aligned}$$

Finally, for a nonzero in $\boldsymbol{z}_k^{n+1}$ when $i = \frac{n}{2}$, we would need

$$\begin{aligned} 2n + 2 - k - \frac{n}{2} + 1 &\leq \frac{n}{2}, \\ n + 3 &\leq k. \end{aligned}$$

As mentioned above, previous vectors in $\hat{\boldsymbol{B}}$ are associated with values of k less than $n-1$. It follows that no vector to the left of this in the matrix has a nonzero in row $p_{\frac{n}{2},\frac{n}{2}}$, and this confirms it is a valid row to extend the embedded upper triangular matrix.

Similarly, it can be shown that the other specified rows, $f(\frac{n}{2}, \frac{n}{2}+2), f(\frac{n}{2}+2, \frac{n}{2}), f(\frac{n}{2}+2, \frac{n}{2}+2)$, are nonzero in the $\boldsymbol{v}_{n-1}^{n+1}$, $\boldsymbol{w}_{n-1}^{n+1}$, and $\boldsymbol{z}_{n-1}^{n+1}$ null vectors, respectively, and are zero for all previous vectors.

Case 2: Assume n is odd. Here the rows to select for the embedded matrix are $f(\frac{n-1}{2}, \frac{n+1}{2})$, $f(\frac{n-1}{2}, \frac{n-1}{2}+2)$, $f(\frac{n+1}{2}+2, \frac{n+1}{2})$, and $f(\frac{n+1}{2}+2, \frac{n-1}{2}+2)$. Again, we consider the first additional row $(p_{\frac{n-1}{2},\frac{n+1}{2}})$ and show that it has a nonzero entry in column $\boldsymbol{u}_{n-1}^{n+1}$ and is zero in all previous columns. From (6.3) we see that

$$\boldsymbol{u}_{n-1}^{n+1}(\tfrac{n-1}{2}, \tfrac{n+1}{2}) = \pm \boldsymbol{T}(n-1, 1) = \pm 1 \neq 0.$$

As in case 1, we use the formulas (6.3)-(6.6) and the variable limits to consider the value of k necessary to have a nonzero value in row $p_{\frac{n-1}{2},\frac{n+1}{2}}$ in the null vectors $\boldsymbol{u}_k^{n+1}, \boldsymbol{v}_k^{n+1}, \boldsymbol{w}_k^{n+1}, \boldsymbol{z}_k^{n+1}$, where $k < n-1$. Substituting the values as in Case 1, we find that for a nonzero in $\boldsymbol{u}_k^{n+1}$, we would need

$$\begin{aligned} \frac{n+1}{2} &\leq k - \frac{n-1}{2} + 1, \\ n &\leq k + 1, \\ n - 1 &\leq k. \end{aligned}$$

For $\boldsymbol{v}_k^{n+1}$, we would need

$$\begin{aligned} \frac{n-1}{2} &\leq \frac{n+1}{2} + k - n - 1, \\ n &\leq k. \end{aligned}$$

For $\boldsymbol{w}_k^{n+1}$, we would need

$$\begin{aligned} \frac{n-1}{2} &\geq \frac{3n+1}{2} - k + 1, \\ n+2 &\leq k. \end{aligned}$$

For $\boldsymbol{z}_k^{n+1}$, we would need

$$\begin{aligned} 2n + 2 - k - \frac{n-1}{2} + 1 &\leq \frac{n+1}{2}, \\ n+2 &\leq k. \end{aligned}$$

It is clear from these bounds that we will not have nonzeros in row $p_{\frac{n-1}{2},\frac{n+1}{2}}$ for null vectors associated with $k < n-1$. Thus, this row is appropriate to add to our embedded matrix. Similarly, the other specified rows, $(\frac{n-1}{2}, \frac{n-1}{2} + 2), (\frac{n+1}{2} + 2, \frac{n+1}{2}), (\frac{n+1}{2} + 2, \frac{n-1}{2} + 2)$, can be shown to fit the necessary criteria to extend the embedded matrix.

Thus, whether n is odd or even, we can choose a subset of rows of $\hat{\boldsymbol{B}}$ such that the selected rows form an upper triangular matrix with nonzero diagonal entries. □

Lemma 6 *The dimension of the null space for a problem of size n is $4(n-1)$.*

Proof: An $n \times n$ problem has n^2 unknown function values and $(n-2)^2$ constraints. The problem matrix, then, has dimension $n^2 \times (n-2)^2$, which gives

$$n^2 - (n-2)^2 = n^2 - (n^2 - 4n + 4) = 4n - 4 = 4(n-1)$$

degrees of freedom by the matrix shape. The minimum dimension of the null space, then, is $4(n-1)$. The original problem matrix is composed of five-point finite-difference-stencil constraints. By inspection, the matrix is upper triangular with a nonzero diagonal. Thus the matrix has full row rank, forcing the maximum size of the null-space basis to be $4(n-1)$.

We conclude that the dimension of the null space is exactly $4(n-1)$. □

Theorem 5 *For even n, $\boldsymbol{Z} = [\boldsymbol{u}_1^n \ \boldsymbol{v}_1^n \ \boldsymbol{w}_1^n \ \boldsymbol{z}_1^n \ \cdots \ \boldsymbol{u}_{n-1}^n \ \boldsymbol{v}_{n-1}^n \ \boldsymbol{w}_{n-1}^n \ \boldsymbol{z}_{n-1}^n]$ is a null basis for our problem.*

Proof: $\boldsymbol{Z} = [\boldsymbol{B} \ \boldsymbol{u}_{n-1}^n \ \boldsymbol{v}_{n-1}^n \ \boldsymbol{w}_{n-1}^n \ \boldsymbol{z}_{n-1}^n]$, where $\boldsymbol{B}$ is as defined in the statement of Lemma 2. By Lemma 2, $\boldsymbol{B}$ has an embedded upper triangular matrix, and thus we have an embedded matrix for the first $4(n-2)$ vectors of $\boldsymbol{Z}$. We

consider the right-most four vectors of $\boldsymbol{Z}$, and show, using an argument similar to that in Lemma 2, that $\boldsymbol{Z}$ also has an embedded upper triangular matrix.

In order to prove that $\boldsymbol{Z}$ has an embedded upper triangular matrix, then, we must show that each of the last four columns has an associated row in which the column has a nonzero value and all previous columns have a zero value. It can be shown that the following rows satisfy those criteria: $f(\frac{n}{2}, \frac{n}{2}), f(\frac{n}{2}, \frac{n}{2} + 1), f(\frac{n}{2} + 1, \frac{n}{2})$, and $f(\frac{n}{2} + 1, \frac{n}{2} + 1)$.

As in Lemma 2, we will prove that the first row above satisfies the necessary criteria. We omit the proofs for the other rows, as the proofs are all very similar. We consider the row $p_{\frac{n}{2},\frac{n}{2}}$. $\boldsymbol{u}^n_{n-1}(\frac{n}{2}, \frac{n}{2}) = \pm\boldsymbol{T}(n-2, 1) = \pm 1 \neq 0$; thus, the nonzero criterion is met. Now we must show that no previous vector is nonzero in this row. We will do this by using formulas (6.3)-(6.6) and their variable limits to consider the value of k necessary to have a nonzero value in row $p_{\frac{n}{2},\frac{n}{2}}$ for the null vectors $\boldsymbol{u}^{n+1}_k$, $\boldsymbol{v}^{n+1}_k$, $\boldsymbol{w}^{n+1}_k$, $\boldsymbol{z}^{n+1}_k$. Substituting $i = \frac{n}{2}$ and $j = \frac{n}{2}$, we find that for a nonzero in $\boldsymbol{u}^n_k$, we would need

$$\begin{aligned} \frac{n}{2} &\leq k - \frac{n}{2} + 1, \\ n - 1 &\leq k. \end{aligned}$$

For a nonzero in $\boldsymbol{v}^n_k$ we would need

$$\begin{aligned} \frac{n}{2} &\leq \frac{n}{2} + k - n, \\ n &\leq k. \end{aligned}$$

For $\boldsymbol{w}^n_k$ we would need

$$\begin{aligned} \frac{n}{2} &\geq \frac{3n}{2} - k, \\ n &\leq k. \end{aligned}$$

For $\boldsymbol{z}^n_k$ we would need

$$\begin{aligned} 2n - k - \frac{n}{2} + 1 &\leq \frac{n}{2}, \\ n + 1 &\leq k. \end{aligned}$$

For vectors to the left of $\boldsymbol{u}^n_{n-1}$, $k < n - 1$. Based on the limits above, then, no previous vector has a nonzero in row $p_{\frac{n}{2},\frac{n}{2}}$, and it fits into our embedded matrix.

The proof follows in a similar manner for the other specified rows.

Thus, $\boldsymbol{Z}$ has an embedded upper triangular matrix, and therefore it has full column rank. The number of columns (and thus the rank) is $4(n-1)$. We know from Theorem 1 that all of the vectors in the matrix are null vectors. By Lemma 3, the null-space basis is size $4(n-1)$. We have a matrix with the correct number of linearly independent null vectors. Thus we have a basis. $\square$

Theorem 6 *For odd n, $\boldsymbol{Z} = [\boldsymbol{B}\ \boldsymbol{u}_{n-1}^n\ \boldsymbol{v}_{n-1}^n\ \boldsymbol{w}_{n-1}^n\ [1, ..., 1]^T]$ is a null basis for our problem.*

Proof: By Lemma 2, $\boldsymbol{B}$ has an embedded upper triangular matrix. Therefore it has full column rank. We consider the right-most four vectors of $\boldsymbol{Z}$ and show, using an argument similar to that of Lemma 2 and Theorem 3, that the last four vectors are independent of the vectors of $\boldsymbol{B}$ and that they are also independent of each other.

Let S be the set of rows $f(\frac{n-1}{2}, \frac{n+1}{2}), f(\frac{n+1}{2}, \frac{n+3}{2}), f(\frac{n+1}{2}, \frac{n-1}{2}), f(\frac{n+1}{2}, \frac{n+1}{2})$. First, we will show, using row $p_{\frac{n-1}{2}, \frac{n+1}{2}}$, that $\boldsymbol{u}_{n-1}^n$ is independent of the columns of $\boldsymbol{B}$.

$$\boldsymbol{u}_{n-1}^n(\tfrac{n-1}{2}, \tfrac{n+1}{2}) = \pm\boldsymbol{T}(n-1, 1) = \pm 1 \neq 0.$$

Thus, $\boldsymbol{u}_{n-1}^n$ has a nonzero value in this row. If no vector of $\boldsymbol{B}$ is nonzero in this row, we have shown linear independence for this case. Similarly to Lemma 2 and Theorem 2, we find that for various vectors to have nonzero values,

$$n - 1 \leq k$$

for $\boldsymbol{u}_k^n$ and $\boldsymbol{v}_k^n$, and

$$n + 1 \leq k$$

for $\boldsymbol{w}_k^n$ and $\boldsymbol{z}_k^n$. All the null vectors of $\boldsymbol{B}$ are formed with $k \leq n - 2$, so they are all zero in this row and thus linearly independent of $\boldsymbol{u}_{n-1}^n$.

Now we consider the column $\boldsymbol{v}_{n-1}^n$ and row $p_{\frac{n+1}{2}, \frac{n+3}{2}}$.

$$\boldsymbol{v}_{n-1}^n(\tfrac{n+1}{2}, \tfrac{n+3}{2}) = \pm\boldsymbol{T}(n-1, 1) = \pm 1 \neq 0.$$

As above, we will show that no vector of $\boldsymbol{B}$ is nonzero in this row. Following the same procedure as previously, we find that for a nonzero value,

$$n + 1 \leq k$$

for $\boldsymbol{u}_k^n$ and $\boldsymbol{w}_k^n$, and

$$n - 1 \leq k$$

for $\boldsymbol{v}_k^n$ and $\boldsymbol{z}_k^n$. Again, we have no nonzeros in this row where $k < n - 1$, and the vectors of $\boldsymbol{B}$ are linearly independent of $\boldsymbol{v}_{n-1}^n$.

Similarly, for column $\boldsymbol{w}_{n-1}^n$ and row $p_{\frac{n+1}{2}, \frac{n-1}{2}}$,

$$\boldsymbol{w}_{n-1}^n(\tfrac{n+1}{2}, \tfrac{n-1}{2}) = \pm\boldsymbol{T}(n-1, 1) = \pm 1 \neq 0,$$

and for a nonzero value in this row

$$n - 1 \leq k$$

for $\boldsymbol{u}_k^n$ and $\boldsymbol{w}_k^n$, and

$$n+1 \leq k$$

for $\boldsymbol{v}_k^n$ and $\boldsymbol{z}_k^n$. We have no nonzeros in this row where $k < n-1$, so the vectors of $\boldsymbol{B}$ are linearly independent of $\boldsymbol{w}_{n-1}^n$.

Finally, for the vector of 1's and row $p_{\frac{n+1}{2},\frac{n+1}{2}}$, $[1 \cdots 1]^T$ has value 1 in position $(\frac{n+1}{2}, \frac{n+1}{2})$, which is nonzero, and for a nonzero value elsewhere in this row,

$$n \leq k$$

for all of $\boldsymbol{u}_k^n$, $\boldsymbol{v}_k^n$, $\boldsymbol{w}_k^n$, and $\boldsymbol{z}_k^n$. Since $k < n-1$ for vectors of $\boldsymbol{B}$, the vectors of $\boldsymbol{B}$ are linearly independent of $[1 \cdots 1]^T$.

We have shown that each of the four right-most vectors of $\boldsymbol{Z}$ is independent of the vectors in $\boldsymbol{B}$. Now we must show that these four vectors are independent of each other. Consider the submatrix formed by the specified rows and these four columns:

$$\begin{bmatrix} 1 & -1 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ -1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

This matrix is the same for all odd values of n up to a factor of -1 applied to each of the first three columns. That factor does not affect linear dependencies. The submatrix does not have an embedded upper triangular matrix, but using Gaussian elimination, we obtain an LU factorization with a $\boldsymbol{U}$ factor of

$$\begin{bmatrix} -1 & 0 & 0 & 1 \\ 0 & -1 & 0 & 1 \\ 0 & 0 & 1 & 3 \\ 0 & 0 & 0 & 1 \end{bmatrix},$$

which demonstrates the linear independence of these four vectors.

Thus, we have a set of $4(n-1)$ linearly independent null vectors ($[1 \cdots 1]^T$ is a null vector by Lemma 1), and therefore we have a basis for our null space. □

6.2.2 Finite Difference 2-D Curl

Next, we consider the curl operator in two dimensions.

For a function $\boldsymbol{y} : \mathbb{R}^2 \to \mathbb{R}^2$,

$$\boldsymbol{y}(t_1, t_2) = \begin{bmatrix} y_1(t_1, t_2) \\ y_2(t_1, t_2) \end{bmatrix},$$

we define the curl as

$$\operatorname{curl} \boldsymbol{y} = \frac{\partial y_2}{\partial t_1} - \frac{\partial y_1}{\partial t_2}.$$

We wish to solve the curl problem

$$\operatorname{curl} \boldsymbol{y} = b,$$

where $b : \mathbb{R}^2 \to \mathbb{R}$ is known, using the null-space method. A null function for the curl operator has two scalar component functions, each of which will be represented in discretized form in seeking an approximate solution.

Given a finite-difference discretization of the curl operator on the unit square in two dimensions, with n nodes in each dimension, the curl operator matrix is of dimension $n^2 \times 2n^2$. This allows for two grids: one for $y_1(t_1, t_2)$ and one for $y_2(t_1, t_2)$. Given no boundary conditions, we cannot evaluate or approximate the finite-difference stencils that extend beyond the boundary of our domain. When we omit those stencils from our matrix, the resulting matrix $\boldsymbol{A}$ has dimension $(n-1)^2 \times 2n^2$. Since $\boldsymbol{A}$ is underdetermined in shape, we know that its null space must be of dimension at least $n^2 + 2n - 1$.

Based on the structure of $\boldsymbol{A}$, we will show how to form a null basis of $\boldsymbol{A}$ explicitly, beginning with a formula for null vectors and following with a proof that we indeed have a null basis. For purposes of the following formulas for null vectors, it is useful to think of null vectors in terms of their numeric values on the finite difference grid. Since the input is a 2-D vector function, we consider the values on two finite-difference grids. We refer to the null vector grids as *null grids*, and we present the explicit formulas for null vectors in terms of values on such grids. Null vectors will be denoted as

$$\boldsymbol{u}_{m,k} = \begin{bmatrix} \boldsymbol{v}_{m,k} \\ \boldsymbol{w}_{m,k} \end{bmatrix},$$

and each of $\boldsymbol{v}$ and $\boldsymbol{w}$ will be described as a grid. The m, k subscripts are used to specify particular null vectors. The $\boldsymbol{v}_{m,k}$ and $\boldsymbol{w}_{m,k}$ null grids are described in terms of values on the finite-difference grid so they will additionally be indexed by rows and columns (e.g., $\boldsymbol{v}_{m,k}(i,j)$) where $1 \le i \le n$ indicates the row and $1 \le j \le n$ indicates the column of the grid point on the unit square, $(0,1)\times(0,1)$, and its boundary. Specifically, $\boldsymbol{v}_{m,k}(i,j)$ corresponds to the value of $\boldsymbol{v}_{m,k}$ at the point $((i-1)h, 1-(j-1)h)$, where $h = 1/(n-1)$. The $(1,1)$ grid point (e.g., $\boldsymbol{v}_{m,k}(1,1)$) corresponds to the top, left entry of the grid. Where possible, we will reserve the use of m and k for null vector subscripts and the use of i and j for grid positions.

There are four types of null vectors. For reasons that will become clear later, we label them out of order here, beginning with type 4 null grids. Type 4 null grids are labeled with $1 \le k \le n$ and $2 \le m \le n$ and are defined as

$$\begin{aligned} \boldsymbol{v}_{m,k}(i,k) &= 1, \qquad 1 \le i \le m-1, \\ \boldsymbol{w}_{m,k}(m,j) &= -1, \qquad 1 \le j \le k. \end{aligned}$$

(This is a total of $n \cdot (n-1) = n^2 - n$ null vectors.) Each of these null grids has a vertical component in $\boldsymbol{v}$ of 1's and a horizontal component in $\boldsymbol{w}$ of -1's.

While the mathematical use of adding these two components together is nonexistent, we do see a nice pattern in these null vectors when we add $\boldsymbol{v}$ and $\boldsymbol{w}$ together. The nonzero components of the sum are L-shaped, i.e., they mark corners in the grid. For example, here are the components for $\boldsymbol{u}_{3,3}$ with $n = 5$:

$$\boldsymbol{v}_{3,3} = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}, \qquad \boldsymbol{w}_{3,3} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ -1 & -1 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}.$$

Combining the two we see the L pattern:

$$\boldsymbol{v}_{3,3} + \boldsymbol{w}_{3,3} = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ -1 & -1 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}.$$

Again, with $\boldsymbol{u}_{2,4}$ for $n = 5$, first the complete null vector:

$$\boldsymbol{v}_{2,4} = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}, \qquad \boldsymbol{w}_{2,4} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ -1 & -1 & -1 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix},$$

and then the sum of the two components, displaying the pattern noted above:

$$\boldsymbol{v}_{2,4} + \boldsymbol{w}_{2,4} = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 \\ -1 & -1 & -1 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}.$$

Next, for type 2 null grids, $m = n + 1$ and $1 \le k \le n - 1$ (total of $n - 1$ null vectors):

$$\boldsymbol{v}_{n+1,k}(i,k) = 1, \qquad 1 \le i \le n.$$

Each of these null grids has a column of ones in the $\boldsymbol{v}$ component and is zero

everywhere in the $\boldsymbol{w}$ component. For example, $\boldsymbol{u}_{6,2}$ for $n = 5$:

$$\boldsymbol{v}_{6,2} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \end{bmatrix}, \qquad \boldsymbol{w}_{6,2} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}.$$

Now, for type 3, $m = 1$ and $1 \leq k \leq n$ (for n additional null vectors):

$$\boldsymbol{w}_{1,k}(1, k) = 1.$$

A null grid of this type has zeros everywhere in the $\boldsymbol{v}$ component and exactly one value of 1 in the $\boldsymbol{w}$ component along the top row of the grid. Here, as an example, is $\boldsymbol{u}_{1,3}$ (with $n = 5$):

$$\boldsymbol{v}_{1,3} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}, \qquad \boldsymbol{w}_{1,3} = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}.$$

And for type 1 null grids, $k = n + 1$ and $1 \leq m \leq n$ (for n additional null vectors):

$$\boldsymbol{v}_{m,n+1}(m, n) = 1.$$

These null grids have a zero $\boldsymbol{w}$ component and exactly one value of 1 in the right column of the $\boldsymbol{v}$ component. An example of this is $\boldsymbol{u}_{5,6}$ (for $n = 5$):

$$\boldsymbol{v}_{5,6} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}, \qquad \boldsymbol{w}_{5,6} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}.$$

Next we will demonstrate that all of these claimed null vectors are actual null vectors. We denote the finite-difference stencils our curl operator creates on our problem grid as $\boldsymbol{g}_{m,k}$, where

$$\boldsymbol{g}_{m,k} = \begin{bmatrix} \boldsymbol{c}_{m,k} \\ \boldsymbol{d}_{m,k} \end{bmatrix}.$$

We have $(n-1)^2 = n^2 - 2n + 1$ stencils interior to our domain. They are defined for $2 \leq m \leq n$ and $1 \leq k \leq n - 1$, with

$$\begin{aligned} \boldsymbol{c}_{m,k}(m-1, k) &= -4, \\ \boldsymbol{c}_{m,k}(m, k) &= 4, \end{aligned}$$

$$\begin{aligned} \boldsymbol{d}_{m,k}(m,k) &= -4, \\ \boldsymbol{d}_{m,k}(m,k+1) &= 4. \end{aligned}$$

Lemma 7 *All of the vectors described above are null vectors.*

Proof: We must show that each null vector gives a weighted sum of zero for each finite-difference stencil. We will consider types of null vectors in the opposite order in which they were presented above, beginning with type 1. Each of these null vectors has exactly one nonzero value, which occurs at $\boldsymbol{v}_{m,n+1}(m,n)$ for any value of m. This will give a nonzero value for the weighted sum if and only if there is a nonzero value in $\boldsymbol{c}_{m,k}(m,n)$ for any finite-difference stencil. But it is clear from the definition of the finite-difference stencils above (since $k \leq n-1$) that the right-most nonzero values in the grid $\boldsymbol{c}_{m,k}$ occur in the $(n-1)$st column. Thus, all of the vectors of type 1 are confirmed as null vectors.

Similarly, the vectors of type 3 will produce a nonzero weighted sum if and only if there is a nonzero value in $\boldsymbol{d}_{m,k}(1,k)$ for any finite-difference stencil. The valid stencils, however, are defined for $m \geq 2$, so all the vectors of type 3 are confirmed as null vectors.

The null vectors of type 2 have a zero $\boldsymbol{w}$ component, so as for type 3 above, we concentrate on the $\boldsymbol{c}$ part of the stencils and how they sum with the $\boldsymbol{v}$ component of our vectors. Each vector $\boldsymbol{v}$ here has one grid column filled with 1's, and 0's elsewhere. Without loss of generality, we select a specific null vector of this type. That is, we select $\boldsymbol{u}_{n+1,p}$ for some p between 1 and $n-1$. The finite-difference stencils with nonzeros overlapping the nonzeros in our vector are those with $k = p$. All others have zero components in the pth grid column and will by construction have a weighted sum of zero. For stencils $\boldsymbol{g}_{m,p}$ we have $\boldsymbol{c}_{m,p}(m-1,p) = -4$ and $\boldsymbol{c}_{m,p}(m,p) = 4$. The weighted sum of our null vector with these stencils will be

$$\begin{aligned} \boldsymbol{v}_{n+1,p}(m-1,p) \cdot \boldsymbol{c}_{m,p}(m-1,p) &+ \boldsymbol{v}_{n+1,p}(m,p) \cdot \boldsymbol{c}_{m,p}(m,p) \\ &= 1 \cdot (-4) + 1 \cdot 4 = -4 + 4 = 0. \end{aligned}$$

This follows for all values of m from 2 to n (which is where the finite-difference stencils are defined), and for each p from 1 to $n-1$. Thus, the vectors of type 2 are indeed null vectors.

Last, we consider the most general type of null vectors above, type 4. Without loss of generality, we select a specific vector $\boldsymbol{u}_{q,p}$ ($m = q$ and $k = p$) within the bounds defined for vectors of type 4. We will show that the vector $\boldsymbol{u}_{q,p}$ satisfies all the finite-difference stencil constraints. With this type of vector, we have several stencils that overlap with the nonzeros in our vector. First, we consider stencils $\boldsymbol{g}_{m,p}$ with $m < q$. With these stencils, we have overlap in the $\boldsymbol{c}$ and $\boldsymbol{v}$ components. In each case, we have the same situation as we had above

for null vectors of type 2 because by definition, all of the values of $\boldsymbol{v}_{q,p}$ are 1 for $m < q$. For any $m < q$, then,

$$\boldsymbol{v}_{q,p}(m-1,p)\cdot\boldsymbol{c}_{m,p}(m-1,p)+\boldsymbol{v}_{q,p}(m,p)\cdot\boldsymbol{c}_{m,p}(m,p) = 1\cdot(-4)+1\cdot 4 = -4+4 = 0.$$

Next, we consider $\boldsymbol{g}_{q,k}$ with $k < p$. Here we have overlap in the $\boldsymbol{d}$ and $\boldsymbol{w}$ components. In each case here we find (for any k, p)

$$\boldsymbol{w}_{q,p}(q,k)\cdot\boldsymbol{d}_{q,k}(q,k)+\boldsymbol{w}_{q,p}(q,k+1)\cdot\boldsymbol{d}_{q,k}(q,k+1) = -1\cdot(-4)+(-1)\cdot 4 = 4-4 = 0,$$

because by definition, all of the values of $\boldsymbol{w}_{q,k}$ are -1 for $k \leq p$.

The last finite-difference stencil that overlaps with our null vector is $\boldsymbol{g}_{q,p}$. This overlaps with components in both grids. The weighted sum here is

$$\begin{aligned} \boldsymbol{v}_{q,p}(q-1,p)\cdot\boldsymbol{c}_{q,p}(q-1,p)+\boldsymbol{v}_{q,p}(q,p)\cdot\boldsymbol{c}_{q,p}(q,p)+ & \\ \boldsymbol{w}_{q,p}(q,k)\cdot\boldsymbol{d}_{q,p}(q,k)+\boldsymbol{w}_{q,p}(q,k+1)\cdot\boldsymbol{d}_{q,p}(q,k+1) &= \\ 1\cdot(-4)+0\cdot 4+(-1)\cdot(-4)+0\cdot 4 &= \\ -4+4 &= 0. \end{aligned}$$

Thus, $\boldsymbol{u}_{q,p}$ has a weighted sum of zero when considered with all finite-difference stencils. This argument holds for any q, p within designated bounds, and in all cases $\boldsymbol{u}_{m,k}$ as a type 4 vector is a valid null vector. □

Now that we have proved that all of the vectors defined above are null vectors, we will show that they are linearly independent. That, together with a proof that the rows of $\boldsymbol{A}$ are independent will give us the final result we desire of showing that these vectors provide a null basis for the curl operator defined above.

First, we define some useful submatrices. Let $\boldsymbol{B_1}$ be the matrix of null vectors of type 1, that is,

$$\boldsymbol{B_1} = [\boldsymbol{u}_{1,n+1}\,\boldsymbol{u}_{2,n+1}\,\cdots\,\boldsymbol{u}_{n,n+1}].$$

Let $\boldsymbol{B_2}$ be the matrix of null vectors of type 2, that is,

$$\boldsymbol{B_2} = [\boldsymbol{u}_{n+1,1}\,\boldsymbol{u}_{n+1,2}\,\cdots\,\boldsymbol{u}_{n+1,n+1}].$$

Let $\boldsymbol{B_3}$ be the matrix of null vectors of type 3, that is,

$$\boldsymbol{B_3} = [\boldsymbol{u}_{1,1}\,\boldsymbol{u}_{1,2}\,\cdots\,\boldsymbol{u}_{1,n}].$$

Let $\boldsymbol{B_4}$ be the matrix of null vectors of type 4, that is,

$$\boldsymbol{B_4} = [\boldsymbol{u}_{2,1}\,\boldsymbol{u}_{2,2}\,\cdots\,\boldsymbol{u}_{2,n}\boldsymbol{u}_{3,1}\,\boldsymbol{u}_{3,2}\,\cdots\,\boldsymbol{u}_{3,n}\,\cdots\boldsymbol{u}_{n,n}].$$

Lemma 8 *For any n, $\boldsymbol{B} = [\boldsymbol{B_1}\ \boldsymbol{B_2}\ \boldsymbol{B_3}\ \boldsymbol{B_4}]$ contains an embedded upper triangular matrix.*

Proof: The matrix $\boldsymbol{B}$ is of dimension $2n^2 \times n^2 + 2n - 1$. To show the existence of an embedded upper triangular matrix, we will show that each column of the matrix has a nonzero value in a matrix row that is zero in each previous matrix column. The rows correspond to grid positions in our double grid; we will refer to the matrix rows in terms of those grid positions, using $\boldsymbol{v}(i,j)$ and $\boldsymbol{w}(i,j)$ as above.

First, we consider $\boldsymbol{B_1}$. By the definition of these null vectors, we know that the matrix has one nonzero in each column. The nonzeros occur in the matrix rows corresponding to grid positions $\boldsymbol{v}(1,n)$ to $\boldsymbol{v}(n,n)$. By the fact that each column has only one nonzero, each matrix row with a nonzero value is zero in that row in all other columns.

Next, we consider the combination of $\boldsymbol{B_1}$ and $\boldsymbol{B_2}$. We know from above that only the matrix rows $\boldsymbol{v}(m,n)$ for $1 \leq m \leq n$ have nonzero values in $\boldsymbol{B_1}$; all the others rows of $\boldsymbol{B_1}$ are entirely zero. All of the rows selected from $\boldsymbol{B_1}$ as part of the embedded matrix are from the right-most grid column, column n, of the grid $\boldsymbol{v}$. The rows we select from $\boldsymbol{B_2}$ to add to our embedded matrix argument are those corresponding to grid positions $\boldsymbol{v}(1,k)$ for $1 \leq k \leq n-1$. Since $k \leq n-1$, these do not conflict with the rows we already selected. Each null vector in $\boldsymbol{B_2}$ has nonzero values in an entire grid column of the null grid $\boldsymbol{v}$. The positions we select for the embedded matrix rows each come from a different grid column. By definition of the null vectors in $\boldsymbol{B_2}$, they do not overlap with each other; each has zeros everywhere in the grid $\boldsymbol{v}$ except the one in which it has 1's. Thus, the selected matrix rows ($\boldsymbol{v}(1,k)$) have nonzeros in exactly one column of $\boldsymbol{B_2}$ (the matrix column corresponding to $\boldsymbol{u}_{n+1,k}$) and zeros in every other column of $\boldsymbol{B_1}$ and $\boldsymbol{B_2}$. It follows that we have an embedded upper triangular matrix in the combined matrix $[\boldsymbol{B_1}\,\boldsymbol{B_2}]$.

Now we append the third matrix to the previous two and consider the matrix $[\boldsymbol{B_1}\,\boldsymbol{B_2}\,\boldsymbol{B_3}]$. The matrix $\boldsymbol{B_3}$ has nonzero values along the top grid row of the the $\boldsymbol{w}$ grid. Similarly to the null vectors that compose $\boldsymbol{B_1}$, these null vectors have exactly one nonzero value. The matrix rows previously selected from $\boldsymbol{B_1}$ and $\boldsymbol{B_2}$ and those corresponding to grid positions $\boldsymbol{w}(1,k)$ for $1 \leq k \leq n$ give us an embedded matrix here. Null vectors of $\boldsymbol{B_1}$ and $\boldsymbol{B_2}$ have nonzeros only in the $\boldsymbol{v}$ grid. These additional matrix rows are zero in $\boldsymbol{B_1}$ and $\boldsymbol{B_2}$ and everywhere in $\boldsymbol{B_3}$ except one matrix column each.

Finally, we put our entire matrix together, and consider the matrix $\boldsymbol{B}$ as a whole. The order of columns within the first three matrices does not matter, but the order of columns within $\boldsymbol{B_4}$ is important to our argument. We order the matrix columns so that they correspond to null vectors $\boldsymbol{u}_{m,k}$, beginning with $\boldsymbol{u}_{2,1}$ and incrementing first by grid columns (k values), then by grid rows (m

values). The order will be as $\boldsymbol{B_4}$ is defined above:

$$\boldsymbol{u}_{2,1}\ \boldsymbol{u}_{2,2}\ \cdots\ \boldsymbol{u}_{2,n}\ \boldsymbol{u}_{3,1}\ \boldsymbol{u}_{3,2}\ \cdots\ \boldsymbol{u}_{3,n}\ \boldsymbol{u}_{4,1}\ \cdots \boldsymbol{u}_{n,n}.$$

Taken in order, each of these null vectors has a nonzero value in the matrix row corresponding to grid position $\boldsymbol{w}(m,k)$. All previously considered null vectors are zero in that row/grid position. Without loss of generality, consider the matrix column of $\boldsymbol{B_4}$ corresponding to the null vector $\boldsymbol{u}_{q,p}$ with $2 \leq q \leq n$ and $1 \leq p \leq n$. By definition, this null vector has nonzero values in grid positions $\boldsymbol{v}(1,p), \boldsymbol{v}(2,p) \cdots, \boldsymbol{v}(q-1,p)$ and $\boldsymbol{w}(q,1), \boldsymbol{w}(q,2), \cdots, \boldsymbol{w}(q,p)$. Only null vectors with $m \leq q$ or $k \leq p$ are to the left of this in $\boldsymbol{B_4}$. These null vectors cannot have a nonzero value in $\boldsymbol{w}_{q,p}$ by definition. All the vectors in $\boldsymbol{B_1}$ and $\boldsymbol{B_2}$ are zero everywhere on the $\boldsymbol{w}$ grid, and the $\boldsymbol{B_3}$ vectors have nonzeros only where $m = 1$, which does not conflict here because $q > 1$. Thus, the matrix column of $\boldsymbol{B_4}$ corresponding to null vector $\boldsymbol{u}_{q,p}$ has the first nonzero value in the row corresponding to the grid position $\boldsymbol{w}(q,p)$. The $\boldsymbol{w}(q,p)$ matrix rows, then, complete the set of rows necessary to describe the embedded matrix in our matrix $\boldsymbol{B}$. □

Lemma 9 *The rows of $\boldsymbol{A}$ are linearly independent.*

Proof: The rows of $\boldsymbol{A}$ are formed by finite difference stencils described earlier in this paper. By inspection, while the matrix is rectangular, its form is upper triangular with nonzero diagonal entries. Thus, the rows are linearly independent. □

Lemma 10 *The null space of $\boldsymbol{A}$ is of dimension $n^2 + 2n - 1$*

Proof: Since the rows of $\boldsymbol{A}$ are linearly independent by Lemma 9, we know the dimension of the null space is based solely on the underdetermined shape of $\boldsymbol{A}$. It follows that the dimension of the null space is the difference between the number of columns of $\boldsymbol{A}$, $2n^2$, and the number of rows of $\boldsymbol{A}$, $(n-1)^2$. Thus, the dimension of the null space is

$$\begin{aligned} 2n^2 - (n-1)^2 &= 2n^2 - (n^2 - 2n + 1) \\ &= 2n^2 - n^2 + 2n - 1 \\ &= n^2 + 2n - 1. \end{aligned}$$

□

Theorem 7 *$\boldsymbol{B}$ is a null basis for $\boldsymbol{A}$*

Proof: From Lemma 7, we know the columns of $\boldsymbol{B}$ are null vectors, and by Lemma 8, we know that $\boldsymbol{B}$ has an embedded upper triangular matrix, which proves that the columns are linearly independent. Counting the columns of $\boldsymbol{B}$,

we have n from $\boldsymbol{B_1}$, $n-1$ from $\boldsymbol{B_2}$, n from $\boldsymbol{B_3}$, and $(n-1)\cdot n$ from $\boldsymbol{B_4}$. Summing these, we have a total of

$$n + n - 1 + n + n(n-1) = n^2 - n + 3n - 1 = n^2 + 2n - 1$$

independent null vectors. Lemma 10 states that the null space is of dimension n^2+2n-1, and $\boldsymbol{B}$ provides us with that many independent null vectors. Thus, $\boldsymbol{B}$ is a basis for the null space of $\boldsymbol{A}$. $\square$

6.2.3 Finite Difference 3-D Curl

Now we advance to three dimensional domains, again considering the curl operator. For a function $\boldsymbol{y} : \mathbb{R}^3 \to \mathbb{R}^3$,

$$\boldsymbol{y}(t_1, t_2, t_3) = \begin{bmatrix} y_1(t_1, t_2, t_3) \\ y_2(t_1, t_2, t_3) \\ y_3(t_1, t_2, t_3) \end{bmatrix},$$

we define the curl as

$$\operatorname{curl} \boldsymbol{y} = \begin{bmatrix} \partial y_3/\partial t_2 - \partial y_2/\partial t_3 \\ \partial y_1/\partial t_3 - \partial y_3/\partial t_1 \\ \partial y_2/\partial t_1 - \partial y_1/\partial t_2 \end{bmatrix}.$$

We wish to solve the curl problem

$$\operatorname{curl} \boldsymbol{y} = \alpha,$$

where $\alpha : \mathbb{R}^3 \to \mathbb{R}^3$ is known, using the null-space method. A null function for the curl operator has three scalar component functions, each of which will be represented in discretized form in seeking an approximate solution.

Given a finite-difference discretization of the curl operator on the unit cube in three dimensions, with n nodes in each dimension, the curl operator matrix is of dimension $3n^3 \times 3n^3$. This allows for three grids: one for $y_1(t_1, t_2, t_3)$, one for $y_2(t_1, t_2, t_3)$, and one for $y_3(t_1, t_2, t_3)$. Given no boundary conditions, we cannot evaluate or approximate the finite-difference stencils that extend beyond the boundary of our domain. When we omit those stencils from the original matrix, the resulting matrix has dimension $3(n-1)^3 \times 3n^3$. Let the matrix $\boldsymbol{A}$ be the discrete form of the curl operator without boundary specifications as just described. Since $\boldsymbol{A}$ is underdetermined in shape, we know that its null space must be of dimension at least $9n^2 - 9n + 3$. In practice, the matrix is also underdetermined numerically, giving a null space of dimension $n^3 + 3n^2 + 3n - 3$.

Based on the structure of $\boldsymbol{A}$, we will show how to form a null basis of $\boldsymbol{A}$ explicitly, beginning with a formula for null vectors and following with a proof that we indeed have a null basis. For purposes of the following formulas for null vectors, it is useful to think of null vectors in terms of their numeric values on

the finite difference grid. Since the input is a 3-D vector function, we consider the values on three finite-difference grids. We refer to the null vector grids as *null grids*, and we present the explicit formulas for null vectors in terms of values on such grids. Null vectors will be denoted as

$$\boldsymbol{u}_{m,p,q} = \begin{bmatrix} \boldsymbol{v}_{m,p,q} \\ \boldsymbol{w}_{m,p,q} \\ \boldsymbol{z}_{m,p,q} \end{bmatrix},$$

and each of $\boldsymbol{v}$, $\boldsymbol{w}$, $\boldsymbol{z}$ will be described as a grid. The m, p, q subscripts are used to specify particular null vectors. The $\boldsymbol{v}_{m,p,q}$, $\boldsymbol{w}_{m,p,q}$, and $\boldsymbol{z}_{m,p,q}$ null grids are described in terms of values on the finite-difference grid so they will additionally be indexed by rows and columns and layers (e.g., $\boldsymbol{v}_{m,p,q}(i,j,k)$) where $1 \leq i \leq n$ indicates the row and $1 \leq j \leq n$ indicates the column and $1 \leq k \leq n$ indicates the layer of the grid point on the unit cube, $(0,1) \times (0,1) \times (0,1)$, and its boundary. Specifically, $\boldsymbol{v}_{m,p,q}(i,j,k)$ corresponds to the value of $\boldsymbol{v}_{m,p,q}$ at the point $((i-1)h, (j-1)h, (k-1)h)$, where $h = 1/(n-1)$. The $(1,1,1)$ grid point (e.g., $\boldsymbol{v}_{m,p,q}(1,1,1)$) corresponds to the lower, left, front entry of the grid. Where possible, we will reserve the use of m, p, and q for null vector subscripts and i, j, and k for grid positions.

There are two types of null vectors, which we refer to as single-point null vectors and six-point null vectors. The single-point null vectors have only one nonzero value, and correspond to the derivative not evaluated for each of the three functions, as we will see later. The single-point null vectors with a nonzero value in the $\boldsymbol{v}$ grid are defined for $1 \leq p \leq n$ and $1 \leq q \leq n$ as

$$\boldsymbol{v}_{n+1,p,q}(n,p,q) = 1$$

and for $1 \leq m \leq n-1$ as

$$\boldsymbol{v}_{m,n+1,n+1}(m,n,n) = 1.$$

Similarly, the single-point null vectors in $\boldsymbol{w}$ are defined for $1 \leq m \leq n$ and $1 \leq q \leq n$ as

$$\boldsymbol{w}_{m,n+1,q}(m,n,q) = 1$$

and for $1 \leq p \leq n-1$ as

$$\boldsymbol{w}_{n+1,p,n+1}(n,p,n) = 1.$$

Finally, the single-point null vectors in $\boldsymbol{z}$ are defined for $1 \leq m \leq n$ and $1 \leq p \leq n$ as

$$\boldsymbol{z}_{m,p,n+1}(m,p,n) = 1$$

and for $1 \leq q \leq n-1$ as

$$\boldsymbol{z}_{n+1,n+1,q}(n,n,q) = 1.$$

The six-point null vectors have a maximum of six nonzero values. They are defined for $1 \leq m \leq n$, $1 \leq p \leq n$, and $1 \leq q \leq n$ as

$$\begin{aligned}
\boldsymbol{v}_{m,p,q}(m,p,q) &= 1 \\
\boldsymbol{v}_{m,p,q}(m-1,p,q) &= -1, \qquad m > 1 \\
\boldsymbol{w}_{m,p,q}(m,p,q) &= 1 \\
\boldsymbol{w}_{m,p,q}(m,p-1,q) &= -1, \qquad p > 1 \\
\boldsymbol{z}_{m,p,q}(m,p,q) &= 1 \\
\boldsymbol{z}_{m,p,q}(m,p,q-1) &= -1, \qquad q > 1
\end{aligned}$$

Next we will demonstrate that these claimed null vectors are actual null vectors. We denote the finite-difference stencils that the curl operator creates on our problem grid as $\boldsymbol{g}_{m,p,q}$, where

$$\boldsymbol{g}_{m,p,q} = \begin{bmatrix} \boldsymbol{c}_{m,p,q} \\ \boldsymbol{d}_{m,p,q} \\ \boldsymbol{f}_{m,p,q} \end{bmatrix}.$$

We have $3(n-1)^3 = 3n^3 - 9n^2 + 9n - 3$ stencils interior to our domain, with $(n-1)^3$ of each of three types of stencils. The stencils are defined for $1 \leq m \leq n-1$, $1 \leq p \leq n-1$, and $1 \leq q \leq n-1$.

The first type of stencils (g^1) have values

$$\begin{aligned}
\boldsymbol{d}_{m,p,q}(m,p,q) &= h, \\
\boldsymbol{d}_{m,p,q}(m,p,q+1) &= -h, \\
\boldsymbol{f}_{m,p,q}(m,p,q) &= -h, \\
\boldsymbol{f}_{m,p,q}(m,p+1,q) &= h.
\end{aligned}$$

All of the finite-difference stencils have a factor of h. We are looking for zero sums of linear combinations with these stencils; the h factor will not have an effect on the sum, so for clarity we omit the h in further discussion. The g^1 stencils, then, have values

$$\begin{aligned}
\boldsymbol{d}_{m,p,q}(m,p,q) &= 1, \\
\boldsymbol{d}_{m,p,q}(m,p,q+1) &= -1, \\
\boldsymbol{f}_{m,p,q}(m,p,q) &= -1, \\
\boldsymbol{f}_{m,p,q}(m,p+1,q) &= 1,
\end{aligned}$$

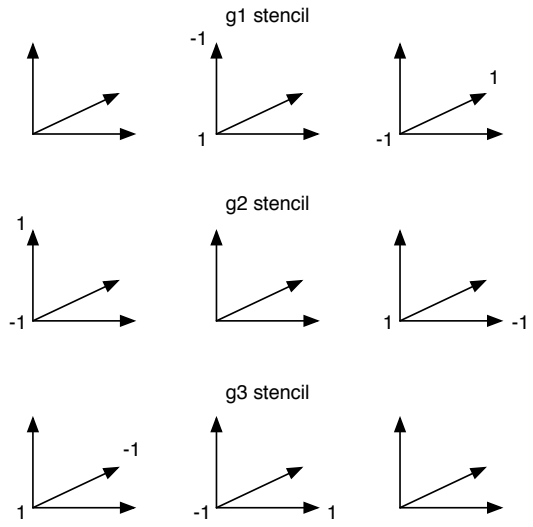

Figure 6.2: Finite difference stencils for 3-D curl

the second type of stencils (g^2) have values

$$\begin{aligned} \boldsymbol{c}_{m,p,q}(m,p,q) &= -1, \\ \boldsymbol{c}_{m,p,q}(m,p,q+1) &= 1, \\ \boldsymbol{f}_{m,p,q}(m,p,q) &= 1, \\ \boldsymbol{f}_{m,p,q}(m+1,p,q) &= -1, \end{aligned}$$

and the third type of stencils (g^3) have values

$$\begin{aligned} \boldsymbol{c}_{m,p,q}(m,p,q) &= 1, \\ \boldsymbol{c}_{m,p,q}(m,p+1,q) &= -1, \\ \boldsymbol{d}_{m,p,q}(m,p,q) &= -1, \\ \boldsymbol{d}_{m,p,q}(m+1,p,q) &= 1. \end{aligned}$$

Figure 6.2 depicts these stencils graphically, where the point (m,p,q) is at the intersection of axes, the t_1 direction is along the horizontal axis, the t_2 direction is along the angled axis that points into the page, and the t_3 direction is along the vertical axis. The graph portions are from $\boldsymbol{c}$, $\boldsymbol{d}$, and $\boldsymbol{f}$ respectively from left to right, and the length of each axis represents the distance of h.

We denote the set of grid points corresponding to the (i,j,k), $(i+1,j,k)$, $(i,j+1,k)$, and $(i,j,k+1)$ points in all of the $\boldsymbol{c}$, $\boldsymbol{d}$, and $\boldsymbol{f}$ functions (total of 12 grid points) as a *mini grid* at (i,j,k). Mini grids are well-defined for $1 \le i \le n-1$, $1 \le j \le n-1$, and $1 \le k \le n-1$.

Lemma 11 *A vector is a null vector for the discrete curl operator described above if and only if the following three conditions are satisfied:*

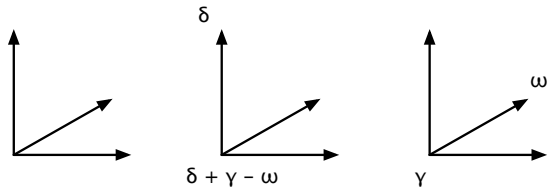

Figure 6.3: Graphical depiction of first condition on null space vectors

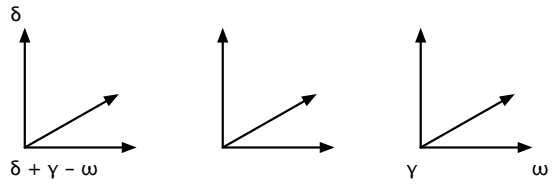

Figure 6.4: Graphical depiction of second condition on null space vectors

- *On each mini grid, the vector values can be described in terms of δ, γ, and ω such that*

$$\begin{aligned} \boldsymbol{w}_{m,p,q}(i,j,k) &= \delta + \gamma - \omega, \\ \boldsymbol{w}_{m,p,q}(i,j,k+1) &= \delta, \\ \boldsymbol{z}_{m,p,q}(i,j,k) &= \gamma, \\ \boldsymbol{z}_{m,p,q}(i,j+1,k) &= \omega, \end{aligned}$$

  as shown in Figure 6.3.

- *On each mini grid, the vector values can be described in terms of δ, γ, and ω such that*

$$\begin{aligned} \boldsymbol{v}_{m,p,q}(i,j,k) &= \delta + \gamma - \omega, \\ \boldsymbol{v}_{m,p,q}(i,j,k+1) &= \delta, \\ \boldsymbol{z}_{m,p,q}(i,j,k) &= \gamma, \\ \boldsymbol{z}_{m,p,q}(i+1,j,k) &= \omega, \end{aligned}$$

  as shown in Figure 6.4.

- *On each mini grid, the vector values can be described in terms of δ, γ, and ω such that*

$$\begin{aligned} \boldsymbol{v}_{m,p,q}(i,j,k) &= \delta + \gamma - \omega, \\ \boldsymbol{v}_{m,p,q}(i,j+1,k) &= \delta, \\ \boldsymbol{w}_{m,p,q}(i,j,k) &= \gamma, \\ \boldsymbol{w}_{m,p,q}(i+1,j,k) &= \omega, \end{aligned}$$

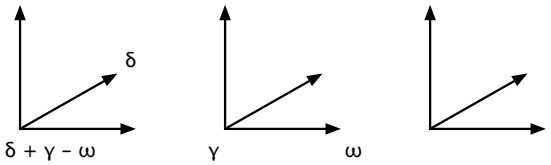

Figure 6.5: Graphical depiction of third condition on null space vectors

as shown in Figure 6.5.

Proof: A null vector must produce a sum of zero when taken as a linear combination with the finite difference stencils of all three types. It suffices to show that a null vector satisfying the forms above satisfies the zero sum constraint for the three stencils on one mini grid because the finite difference stencils are the same for every mini grid.

Assume we have a vector $\boldsymbol{u} = [\boldsymbol{v}\ \boldsymbol{w}\ \boldsymbol{z}]^T$ that satisfies the conditions above. To show it is a null vector, we demonstrate that it satisfies each of the three constraints given by the finite-difference stencils, beginning with the first. Since $\boldsymbol{u}$ satisfies the first condition above, the linear combination given by the first constraint (centered at (i, j, k)) simplifies to

$$\begin{aligned} d(i,j,k) \cdot w(i,j,k) + d(i,j,k+1) \cdot w(i,j,k+1) + \\ f(i,j,k) \cdot z(i,j,k) + f(i,j+1,k) \cdot z(i,j+1,k) &= \\ 1 \cdot (\delta + \gamma - \omega) - 1 \cdot \delta - 1 \cdot \gamma + 1 \cdot \omega &= \\ (\delta - \delta) + (\gamma - \gamma) + (-\omega + \omega) &= 0. \end{aligned}$$

Thus, the g^1 constraint is satisfied. Next we consider the second finite-difference constraint and use the fact that our vector $\boldsymbol{u}$ meets the requirements of the second condition above. Here the linear combination is

$$\begin{aligned} c(i,j,k) \cdot v(i,j,k) + c(i,j,k+1) \cdot v(i,j,k+1) + \\ f(i,j,k) \cdot z(i,j,k) + f(i+1,j,k) \cdot z(i+1,j,k) &= \\ -1 \cdot (\delta + \gamma - \omega) + 1 \cdot \delta + 1 \cdot \gamma - 1 \cdot \omega &= \\ (-\delta + \delta) + (-\gamma + \gamma) + (\omega - \omega) &= 0, \end{aligned}$$

again simplifying to zero as desired. Similarly, the third constraint is met because $\boldsymbol{u}$ satisfies the third condition above:

$$\begin{aligned} c(i,j,k) \cdot v(i,j,k) + c(i,j+1,k) \cdot v(i,j+1,k) + \\ d(i,j,k) \cdot w(i,j,k) + d(i+1,j,k) \cdot w(i+1,j,k) &= \\ 1 \cdot (\delta + \gamma - \omega) - 1 \cdot \delta - 1 \cdot \gamma + 1 \cdot \omega &= \\ (\delta - \delta) + (\gamma - \gamma) + (-\omega + \omega) &= 0. \end{aligned}$$

Thus, $\boldsymbol{u}$ is a null vector if it satisfies the conditions above.

Now, assuming $\boldsymbol{u}$ is a null vector, we will show it must satisfy the conditions above. Since $\boldsymbol{u}$ is a null vector, it gives a zero sum for every linear combination with the finite difference stencils. Using the g^1 stencil, we know

$$\boldsymbol{w}(i,j,k) - \boldsymbol{w}(i,j,k+1) - \boldsymbol{z}(i,j,k) + \boldsymbol{z}(i,j+1,k) = 0$$

for $1 \le i \le n-1$, $1 \le j \le n-1$, and $1 \le k \le n-1$. For ease of notation, let

$$\begin{aligned} \boldsymbol{w}(i,j,k) &= x_1, \\ \boldsymbol{w}(i,j,k+1) &= x_2, \\ \boldsymbol{z}(i,j,k) &= x_3, \\ \boldsymbol{z}(i,j+1,k) &= x_4. \end{aligned}$$

Rewriting the equation, $x_1 - x_2 - x_3 + x_4 = 0$. We know a solution to this equation, namely, $x_1 = x_2 = x_3 = x_4 = 0$, but that is not the only solution. If we combine our variables into a vector $\boldsymbol{x} = [x_1 \; x_2 \; x_3 \; x_4]^T$, a bit of linear algebra shows that a null basis for this sub-problem is

$$\begin{bmatrix} 1 & 1 & -1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

We can add any vector in the null space to our specific solution to obtain another solution. In addition, any vector solving this problem can be written in terms of the specific solution and a linear combination of the null basis vectors. That is, any solution is of the form

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} + \delta \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \end{bmatrix} + \gamma \begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \end{bmatrix} + \omega \begin{bmatrix} -1 \\ 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} \delta + \gamma - \omega \\ \delta \\ \gamma \\ \omega \end{bmatrix}.$$

Similarly, from the g^2 and g^3 stencils, we find that the second and third conditions above must hold. □

Lemma 12 *The six-point vectors described above are null vectors.*

Proof: Each six-point vector has nonzero values on ten mini grids. Specifically, $\boldsymbol{u}_{m,p,q}$ has nonzero values on the mini grids at (m,p,q), $(m-1,p,q)$, $(m,p-1,q)$, $(m,p,q-1)$, $(m-1,p-1,q)$, $(m-1,p,q-1)$, $(m,p-1,q-1)$, $(m-2,p,q)$, $(m,p-2,q)$, $(m,p,q-2)$. throughout the domain. On seven of these mini grids the nonzero values of the six-point vector occur at grid points with nonzero values of the finite-difference stencil on the mini grid. We now

show that the three conditions of Lemma 11 are met for each of these seven mini grids by giving values for δ, γ, and ω in each case.

- Mini grid at (m, p, q)
  1. $\gamma = 1$, $\delta = \omega = 0$
  2. $\gamma = 1$, $\delta = \omega = 0$
  3. $\gamma = 1$, $\delta = \omega = 0$
- Mini grid at $(m-1, p, q)$
  1. $\delta = \gamma = \omega = 0$
  2. $\omega = 1$, $\delta = \gamma = 0$
  3. $\omega = 1$, $\delta = \gamma = 0$
- Mini grid at $(m, p-1, q)$
  1. $\omega = 1$, $\delta = \gamma = 0$
  2. $\delta = \gamma = \omega = 0$
  3. $\delta = 1$, $\gamma = -1$, $\omega = 0$
- Mini grid at $(m, p, q-1)$
  1. $\delta = 1$, $\gamma = -1$, $\omega = 0$
  2. $\delta = 1$, $\gamma = -1$, $\omega = 0$
  3. $\delta = \gamma = \omega = 0$
- Mini grid at $(m-1, p-1, q)$
  1. $\delta = \gamma = \omega = 0$
  2. $\delta = \gamma = \omega = 0$
  3. $\delta = \omega = -1$, $\gamma = 0$
- Mini grid at $(m-1, p, q-1)$
  1. $\delta = \gamma = \omega = 0$
  2. $\delta = \omega = -1$, $\gamma = 0$
  3. $\delta = \gamma = \omega = 0$
- Mini grid at $(m, p-1, q-1)$
  1. $\delta = \omega = -1$, $\gamma = 0$
  2. $\delta = \gamma = \omega = 0$
  3. $\delta = \gamma = \omega = 0$

On every other mini grid, the conditions are met with $\gamma = \delta = \omega = 0$. Thus, by Lemma 11, the six-point vectors are null vectors. $\square$

Lemma 13 *The single-point vectors described above are null vectors.*

Proof: In every case, by construction, the single-point null vectors have one nonzero value at a point on the complete grid where all finite-difference stencils are zero. Thus, every finite-difference constraint will yield a zero sum value. □

Let matrix $\boldsymbol{B}$ of dimension $3n^3 \times (n^3 + 3n^2 + 3n - 5)$ consist of columns of all of the one-point null vectors and all of the six-point null vectors except for $\boldsymbol{u}_{n,n,n-1}$ and $\boldsymbol{u}_{n,n,n}$.

Lemma 14 *Matrix $\boldsymbol{B}$ has linearly independent columns.*

Proof: One way to prove linear independence of columns is to demonstrate that $\boldsymbol{A}$ has an embedded upper triangular matrix. In our context, each grid point corresponds to a row of the matrix, so it suffices to give an ordering of null vectors such that the $(i+1)$st vector has a nonzero value at a grid point where vectors $1, 2, \ldots, i$ are zero. As we add vectors, grid points that have a nonzero in any previous vectors will be called "busy," while grid points that have a value of zero for all previous vectors will be called "free."

The single-point vectors are ordered first. Each has a unique position of one nonzero value, so we know they are linearly independent. Thus the nodes on the right face and the top, back edge of $\boldsymbol{c}$, the nodes on the back face and the top, right edge of $\boldsymbol{d}$, and the nodes on the top face and the back, right edge of $\boldsymbol{f}$ become busy.

Next, we add the six-point vectors $\boldsymbol{u}_{m,p,q}$ for $1 \le m \le n-1$, $1 \le p \le n-1$, and $1 \le q \le n-1$, starting with $\boldsymbol{u}_{1,1,1}$ and incrementing m, p, and q in order. That is, first we add $\boldsymbol{u}_{1,1,1}$, which has a nonzero at the previously free grid point $\boldsymbol{d}(1,1,1)$. Next, we add $\boldsymbol{u}_{2,1,1}$, which has a nonzero at the previously free grid point $\boldsymbol{d}(2,1,1)$, and so on, incrementing the m values up to $n-1$, then incrementing the p value and repeating, i.e., the order is $\boldsymbol{u}_{1,1,1}$, $\boldsymbol{u}_{2,1,1}$, …, $\boldsymbol{u}_{n-1,1,1}$, $\boldsymbol{u}_{1,2,1}$, …, $\boldsymbol{u}_{n-1,2,1}$, $\boldsymbol{u}_{1,3,1}$, …. This ordering is important because it leaves the $\boldsymbol{d}(m,p,q)$ grid position free until the $\boldsymbol{u}_{m,p,q}$ six-point vector is encountered as we proceed from left to right through the columns of $\boldsymbol{B}$. After adding $(n-1)^3$ six-point vectors, the grid points with $1 \le m \le n-1$, $1 \le p \le n-1$, and $1 \le q \le n-1$, are busy in all three grids, $\boldsymbol{c}$, $\boldsymbol{d}$, and $\boldsymbol{f}$.

We have yet to order the six-point vectors with center nodes on the right, back, and top faces of our domain. We begin ordering that set by adding the $\boldsymbol{u}_{m,n,q}$ six-point vectors with $1 \le m \le n-1$, and $1 \le q \le n-1$, which cover the free nodes $\boldsymbol{f}(m,n,q)$, and the $\boldsymbol{u}_{m,n,n}$ six-point vectors for $1 \le m \le n-1$, which cover free nodes $\boldsymbol{d}(m,n-1,n)$. Similarly, we add the $\boldsymbol{u}_{m,p,n}$ six-point vectors for $1 \le m \le n-1$ and $1 \le p \le n-1$, busying the free nodes $\boldsymbol{c}(m,p,n)$. Then we add the $\boldsymbol{u}_{n,p,n}$ vectors with $1 \le p \le n-1$. These claim the $\boldsymbol{f}(n,p,n-1)$ nodes. Next in order are the $\boldsymbol{u}_{n,p,q}$ six-point vectors with $1 \le p \le n-1$ and $1 \le q \le n-1$. These cover the free nodes $\boldsymbol{d}(n,p,q)$.

At this point, all the nodes in every grid are busy. We have shown that an embedded upper triangular matrix exists amongst these first $n^3 + 3n^2 + 2n - 3$ column vectors, and thus have proved that they are linearly independent. For the last set of null vectors, the $\boldsymbol{u}_{n,n,q}$ six-point vectors for $1 \leq q \leq n-2$, we use a proof by contradiction to show they are independent.

Let the null vectors considered so far be columns of the matrix $\hat{\boldsymbol{B}}$. We begin with the null vector $\boldsymbol{x} = \boldsymbol{u}_{n,n,1}$. Suppose $\boldsymbol{x}$ is in the span of $\hat{\boldsymbol{B}}$, i.e., adding $\boldsymbol{x}$ to the set of null vectors results in a linearly dependent set. Then,

$$\hat{\boldsymbol{B}}\boldsymbol{y} = \boldsymbol{x}$$

for some vector $\boldsymbol{y}$.

By inspection, at most two null vectors have a nonzero value at each grid point, allowing us to back-substitute easily. First, we note that the vector values of $\boldsymbol{y}$ corresponding to the null vectors $\boldsymbol{u}_{n+1,n+1,1}$, $\boldsymbol{u}_{n,n+1,1}$, $\boldsymbol{u}_{n+1,n,1}$ must be equal to 1. (These are one-point null vectors.) Similarly, the elements of $\boldsymbol{y}$ corresponding to the null vectors $\boldsymbol{u}_{n-1,n,1}$ and $\boldsymbol{u}_{n,n-1,1}$ (six-point null vectors) must have the value -1. And so the nonzero values of $\boldsymbol{x}$ are attainable. In addition, however, every additional value must be zero. Continuing the back-substitution for the grid values of $\boldsymbol{c}$, we first find that the values of $\boldsymbol{y}$ corresponding to the null vectors $\boldsymbol{u}_{m,n,1}$ and $\boldsymbol{u}_{m,n-1,1}$ $(1 \leq m \leq n-1)$ must be -1. Then, looking at the nodes of $\boldsymbol{d}$, the values of $\boldsymbol{y}$ corresponding to the null vectors $\boldsymbol{u}_{n-1,p,1}$ must be -1. Following this pattern through the $\boldsymbol{f}$ grid and back again, we find we cannot zero out the values at grid points $\boldsymbol{c}(n-1,n,k)$ and $\boldsymbol{d}(n,n-1,k)$ for $2 \leq k \leq n-1$. Thus, we have a contradiction and $\boldsymbol{x}$ is independent of the previous null vectors.

This process can be repeated sequentially for null vectors $\boldsymbol{u}_{n,n,q}$ for $2 \leq q \leq n-2$. At each step, the nonzero values of the linear combination are $\boldsymbol{c}(n-1,n,k)$ and $\boldsymbol{d}(n,n-1,k)$ for $q+1 \leq k \leq n-1$. Thus, the complete set of columns of $\boldsymbol{B}$ is linearly independent. □

Lemma 15 *At least* $2n^3 - 2n^2 - 3n - 1$ *rows of* $\boldsymbol{A}$ *are linearly independent.*

Proof: One way to prove linear independence of rows is to demonstrate that $\boldsymbol{A}$ has an embedded upper triangular matrix with the correct number of columns. In our context, each grid point corresponds to a column of the matrix, so it suffices to give an ordering of finite-difference stencils (corresponding to rows in $\boldsymbol{A}$) such that the $(i+1)$st stencil has a nonzero value at a grid point where stencils $1, 2, \ldots, i$ are zero. We next explain such an ordering that shows several rows $(2(n-1)^3 + (n-1)^2 + (n-1)\cdot(n-2)\ )$ to be independent. Later we show the last $(n-2)^2$ rows are independent using a proof by contradiction. For reference, the finite difference stencils are displayed in Figure 6.2.

To begin, we order all the $\boldsymbol{g}^1$ stencils. Any consistent ordering in each of the three variables will show an embedded matrix structure. We will order them in

increasing order according to the first variable, then the second variable, and then the third variable. That is, the ordering is $\boldsymbol{g}^1_{1,1,1}$, $\boldsymbol{g}^1_{2,1,1}$, $\ldots$, $\boldsymbol{g}^1_{n-1,1,1}$, $\boldsymbol{g}^1_{1,2,1}$, $\ldots \boldsymbol{g}^1_{n-1,2,1}$, $\boldsymbol{g}^1_{1,3,1}$, $\ldots$, $\boldsymbol{g}^1_{1,1,2}$, $\ldots$. With this ordering, each stencil $\boldsymbol{g}^1_{i,j,k}$ covers the grid position $\boldsymbol{d}(i,j,k+1)$ for the first time.

Next are all the $\boldsymbol{g}^2$ stencils, which we order the same way as the $\boldsymbol{g}^1$ stencils. Each $\boldsymbol{g}^2_{i,j,k}$ stencil covers the grid position $\boldsymbol{c}(i,j,k+1)$ for the first time. Together, the $\boldsymbol{g}^1$ and $\boldsymbol{g}^2$ stencils provide us with $2(n-1)^3$ linearly independent rows.

Now we add the $\boldsymbol{g}^3_{i,n-1,k}$ stencils for $1 \le i \le n-1$ and $1 \le k \le n-1$. These are added in any order and each covers the previously uncovered $\boldsymbol{c}(i,n,k)$ grid position. After those, we add the $\boldsymbol{g}^3_{n-1,j,k}$ stencils for $1 \le j \le n-2$ and $1 \le k \le n-1$. These are also added in any order and each covers the $\boldsymbol{d}(n,j,k)$ grid position for the first time. We now have every grid point covered that can be by the finite difference stencils, and we have shown $2(n-1)^3+(n-1)^2+(n-1)\cdot(n-2)$ vectors to be linearly independent.

The $\boldsymbol{g}^3_{1,j,k}$ stencils (for $1 \le j \le n-2$ and $1 \le k \le n-2$) can also be added to the set without creating a dependency between rows, as we now show by contradiction. We add these vectors in order of descending indices, first in k, then in j. The order is $\boldsymbol{g}^3_{1,n-2,n-2}$, $\boldsymbol{g}^3_{1,n-2,n-3}$, $\boldsymbol{g}^3_{1,n-2,n-4}$, $\ldots$, $\boldsymbol{g}^3_{1,n-2,1}$, $\boldsymbol{g}^3_{1,n-3,n-1}$, $\ldots$.

Suppose the stencil $\boldsymbol{g}^3_{1,p,q}$ ($1 \le p,q \le n-2$) is dependent on other rows in the set, that is, the rows created by the $\boldsymbol{g}^1_{i,j,k}$ ($1 \le i,j,k \le n-1$), $\boldsymbol{g}^2_{i,j,k}$ ($1 \le i,j,k \le n-1$), $\boldsymbol{g}^3_{i,n-1,k}$ ($1 \le i,k \le n-1$), $\boldsymbol{g}^3_{n-1,j,k}$ ($1 \le j \le n-2$, $1 \le k \le n-1$), and $\boldsymbol{g}^3_{1,j,k}$ ($j > p$, $1 \le k \le n-2$ and $j = p$, $k > q$) stencils. Then, for some values of γ_e,

$$\begin{aligned}
\boldsymbol{g}^3_{1,p,q} \;=\; & \sum_{e=1}^{(n-1)^3} \gamma_e \boldsymbol{g}^1_{i,j,k} + \sum_{e=(n-1)^3+1}^{2(n-1)^3} \gamma_e \boldsymbol{g}^2_{i,j,k} + \sum_{e=2(n-1)^3+1}^{2(n-1)^3+(n-1)^2} \gamma_e \boldsymbol{g}^3_{i,n-1,k} \\
& + \sum_{e=2(n-1)^3+(n-1)^2+1}^{2(n-1)^3+(n-1)^2+(n-1)(n-2)} \gamma_e \boldsymbol{g}^3_{n-1,j,k} \\
& + \sum_{e=2(n-1)^3+(n-1)(2n-3)+1}^{2(n-1)^3+(n-1)(2n-3)+(n-2)(n-2-p)+(n-2-q)} \gamma_e \boldsymbol{g}^3_{1,j,k}.
\end{aligned}$$

This equality holds at each grid point. Since $\boldsymbol{g}^3_{1,p,q}$ has a value of 1 at $\boldsymbol{c}(1,p,q)$, γ_e corresponding to $\boldsymbol{g}^2_{1,p,q}$ must also equal 1 because $\boldsymbol{g}^2_{1,p,q}$ is the only other stencil in our set with a nonzero at that grid point. Then, since the coefficient of $\boldsymbol{g}^2_{1,p,q}$ is equal to 1, the coefficient of $\boldsymbol{g}^2_{2,p,q}$ must be 1 to give a linear combination of 0 at the grid point $\boldsymbol{c}(2,p,q)$. Similarly, all the coefficients of $\boldsymbol{g}^2_{i,p,q}$ must be 1 for all $1 \le i \le n-1$. But, since the coefficient of $\boldsymbol{g}^2_{n-1,p,q}$ is 1, the linear combination at grid point $\boldsymbol{c}(n,p,q)$ is equal to 1 while the value of $\boldsymbol{g}^3_{1,p,q}$ is 0 at $\boldsymbol{c}(n,p,q)$. Thus, the equality above does not hold and we have a contradiction. It follows that $\boldsymbol{g}^3_{1,p,q}$ is linearly independent from other rows in the set.

Adding this last set of $\boldsymbol{g}^3$ stencils, we have $2(n-1)^3+(n-1)^2+(n-1)(n-2)+(n-2)^2 = 2n^3-2n^2-3n-1$ independent rows. $\square$

Lemma 16 *The null space of $\boldsymbol{A}$ is of dimension at most n^3+3n^2+3n-5.*

Proof: Matrix $\boldsymbol{A}$ has dimension $3(n-1)^3 \times 3n^3$, so by shape the null space has dimension at least

$$3n^3-3(n-1)^3 = 3n^3-3(n^3-3n^2+3n-1) = 3n^3-3n^3+9n^2-9n+3 = 9n^2-9n+3.$$

The maximum dimension of $N(\boldsymbol{A})$ occurs when the fewest rows of $\boldsymbol{A}$ are independent. From Lemma 15, we know that $2n^3-2n^2-3n-1$ rows are independent, leaving the possibility that $3n^3-2n^3-2n^2-3n-1 = (n-2)^3$ are dependent. The maximum dimension of $N(\boldsymbol{A})$, then, is $9n^2-9n+3+(n-2)^3 = 9n^2-9n+3+n^3-6n^2+12n-8 = n^3+3n^2+3n-5$. $\square$

Theorem 8 *Matrix $\boldsymbol{B}$ is a null basis for $\boldsymbol{A}$*

Proof: By Lemma 12 and Lemma 13 all the vectors in $\boldsymbol{B}$ are null vectors of $\boldsymbol{A}$. By Lemma 15, all of the columns are linearly independent. Counting the number of columns, we have $3(n^2+n-1) = 3n^2+3n-3$ columns of single-point null vectors and n^3-2 columns of six-point null vectors for a total of n^3+3n^2+3n-5 columns. The maximum dimension of the null basis is n^3+3n^2+3n-5 by Lemma 16. Therefore, since $\boldsymbol{B}$ is a matrix of n^3+3n^2+3n-5 independent null vectors of $\boldsymbol{A}$, $\boldsymbol{B}$ is a null basis for $\boldsymbol{A}$. $\square$

6.2.4 Bilinear Finite Element 3-D Curl

Here we consider the same operator as the previous section, but we use a Galerkin finite-element discretization with bilinear basis functions. For a function $\boldsymbol{y}: \mathbb{R}^3 \to \mathbb{R}^3$,

$$\boldsymbol{y}(t_1,t_2,t_3) = \begin{bmatrix} y_1(t_1,t_2,t_3) \\ y_2(t_1,t_2,t_3) \\ y_3(t_1,t_2,t_3) \end{bmatrix},$$

we define the curl as

$$\operatorname{curl} \boldsymbol{y} = \begin{bmatrix} \partial y_3/\partial t_2 - \partial y_2/\partial t_3 \\ \partial y_1/\partial t_3 - \partial y_3/\partial t_1 \\ \partial y_2/\partial t_1 - \partial y_1/\partial t_2 \end{bmatrix}.$$

We wish to solve the curl problem

$$\operatorname{curl} \boldsymbol{y} = \alpha,$$

where $\alpha: \mathbb{R}^3 \to \mathbb{R}^3$ is known, using the null-space method. A null function for the curl operator has three scalar component functions, each of which will be represented in discretized form in seeking an approximate solution.

Given a Galerkin finite-element discretization of the curl operator on the unit cube in three dimensions, with n uniformly-spaced nodes in each dimension, the curl operator matrix is of dimension $3n^3 \times 3n^3$. This allows for three linear combination approximations: $y_1(t_1, t_2, t_3) = \mathbf{\Phi} x_1$, $y_2(t_1, t_2, t_3) = \mathbf{\Phi} x_2$, and $y_3(t_1, t_2, t_3) = \mathbf{\Phi} x_3$. Without boundary conditions, we cannot evaluate or constrain the basis functions that are nonzero on the boundary of our domain. When we omit those constraints from the original matrix, the resulting matrix has dimension $3(n-2)^3 \times 3n^3$. Let the matrix $\boldsymbol{A}$ be the discrete form of the curl operator without boundary specifications as just described. Since $\boldsymbol{A}$ is underdetermined in shape, we know that its null space must be of dimension at least $6n^2 - 12n + 24$. In practice, the matrix has less than full row rank, giving a null space of dimension $n^3 + 6n^2 + 12n - 40$.

We will show how to form a null basis of $\boldsymbol{A}$ explicitly by giving formulas for three types of null vectors and identifying which combination of those is a linearly independent set for a uniform discretization with n nodes in each direction. For understanding the following formulas for null vectors, it is useful to think of null vectors in terms of their numeric values on the grid created by discretization. The null vector gives coefficients for a 3-D *vector* function, and we describe the null vector values for the three solution functions on separate grids. We refer to the null vector grids as *null grids*, and we present the explicit formulas for null vectors in terms of values on such grids. Each value on the grid is the coefficient for the basis function that is nonzero at that particular grid point. Null vectors will be denoted as

$$\boldsymbol{u}_{m,p} = \begin{bmatrix} \boldsymbol{v}_{m,p} \\ \boldsymbol{w}_{m,p} \\ \boldsymbol{z}_{m,p} \end{bmatrix},$$

and each of $\boldsymbol{v}$, $\boldsymbol{w}$, $\boldsymbol{z}$ will be described as a grid. The m, p subscripts are used to specify particular null vectors. The $\boldsymbol{v}_{m,p}$, $\boldsymbol{w}_{m,p}$, and $\boldsymbol{z}_{m,p}$ null grids are described in terms of values on the problem grid, so they will be additionally indexed by rows and columns and layers (e.g., $\boldsymbol{v}_{m,p}(i, j, k)$), where $1 \leq i \leq n$ indicates the row, $1 \leq j \leq n$ indicates the column, and $1 \leq k \leq n$ indicates the layer of the grid point in the unit cube, $(0, 1) \times (0, 1) \times (0, 1)$ and its boundary. Specifically, $\boldsymbol{v}_{m,p}(i, j, k)$ corresponds to the value of $\boldsymbol{v}_{m,p}$ at the point $((i-1)h, (j-1)h, (k-1)h)$, where $h = 1/(n-1)$. The $(1, 1, 1)$ grid point (e.g., $\boldsymbol{v}_{m,p}(1, 1, 1)$) corresponds to the lower, left, front entry of the grid.

The first type of null vector, which we denote as $\boldsymbol{u}_{1,p}$, has a line of nonzero values in each null grid. The three lines originate from a common grid point, and the values follow the sequence $s_1(1) = 1$, $s_1(2) = -4$, $s_1(3) = 14$, $s_1(r) = -s_1(r-2) - 4s_1(r-1)$ for $r \geq 4$. The first several values of the sequence are: 1, -4, 14, -52, 194, -724, 2702, -10084. Specifically, for $p = (k-1)n^2 + (j-1)n + i$,

$(1 \le i \le n,\ 1 \le j \le n,\ 1 \le k \le n)$,

$$\begin{aligned} \boldsymbol{v}_{1,p}(r,j,k) &= s_1(i-r+1), \quad r = 1,\ldots,i, \\ \boldsymbol{w}_{1,p}(i,r,k) &= s_1(j-r+1), \quad r = 1,\ldots,j, \\ \boldsymbol{z}_{1,p}(i,j,r) &= s_1(k-r+1), \quad r = 1,\ldots,k. \end{aligned}$$

The second type of null vector uses values from a similar sequence, merely removing the third starting value from s_1 and restarting the recurrence. Here, $s_2(1) = 1$, $s_2(2) = -4$, and $s_2(r) = -s_2(r-2) - 4s_2(r-1)$ for $r \ge 3$. The first several values of the sequence are: 1, -4, 15, -56, 209, -780, 2911, -10864. These null vectors have nonzeros in one function at a time. They are characterized by a line of nonzero values extending from one side of the domain almost to the other side. The $\boldsymbol{u}_{2,p}$ null vectors are of the second type and have nonzero coefficients over the domain of the first function. For $p = 2(k-1)n + 2j - 1$, $(1 \le j \le n,\ 1 \le k \le n)$,

$$\boldsymbol{v}_{2,p}(r,j,k) = s_2(n-r), \quad r = 1,\ldots,n-1,$$

and for $p = 2(k-1)n + 2j$, $(1 \le j \le n,\ 1 \le k \le n)$,

$$\boldsymbol{v}_{2,p}(r,j,k) = s_2(r), \quad r = 2,\ldots,n.$$

Similarly, the null vectors of the second kind with nonzeros in the second function are defined to be

$$\boldsymbol{w}_{3,p}(i,r,k) = s_2(n-r), \quad r = 1,\ldots,n-1,$$

for $p = 2(k-1)n + 2i - 1$, $(1 \le i \le n,\ 1 \le k \le n)$, and

$$\boldsymbol{w}_{3,p}(i,r,k) = s_2(r), \quad r = 2,\ldots,n,$$

for $p = 2(k-1)n + 2i$, $(1 \le i \le n,\ 1 \le k \le n)$. Not surprisingly, we also have a set of null vectors of this variety with nonzero values in the third function.

$$\boldsymbol{z}_{4,p}(i,j,r) = s_2(n-r), \quad r = 1,\ldots,n-1,$$

for $p = 2(j-1)n + 2i - 1$, $(1 \le i \le n,\ 1 \le j \le n)$, and

$$\boldsymbol{z}_{4,p}(i,j,r) = s_2(r), \quad r = 2,\ldots,n,$$

for $p = 2(j-1)n + 2i$, $(1 \le i \le n,\ 1 \le j \le n)$.

The last type of null vector we use to create an independent set uses a sequence similar to that used in the second type. In fact, the sequence is built using the values of s_2. Here we use $s_3(1) = 1$, $s_3(2) = -4$, and $s_3(r) = s_2(r) + s_3(r-2)$. The first several values are: 1, -4, 16, -60, 225, -840, 3136,

-11704. Writing out the recursion, this third sequence is equivalent to the sum of the odd terms or even terms of the s_2 sequence. The Online Encyclopedia of Integer Sequences [48] contains this sequence without the sign changes as the expansion of a polynomial $1/((1-t^2)(1-4t+t^2))$. These values are arranged diagonally, spreading from two opposite corners on a particular layer in the grid. The values of this layer in one case are given by an $n \times n$ grid $\boldsymbol{\ell}_1$, with

$$\begin{aligned} \boldsymbol{\ell}_1(i,j) &= s_3(i-j), \quad 1 < j < n-1, \quad j+1 < i < n, \\ \boldsymbol{\ell}_1(i,j) &= s_3(j-2-i), \quad 4 < j < n, \quad 1 < i < j-3. \end{aligned}$$

For $n = 6$,

$$\boldsymbol{\ell}_1 = \begin{bmatrix} 0 & 0 & 0 & 1 & -4 & 16 \\ 1 & 0 & 0 & 0 & 1 & -4 \\ -4 & 1 & 0 & 0 & 0 & 1 \\ 16 & -4 & 1 & 0 & 0 & 0 \\ -60 & 16 & -4 & 1 & 0 & 0 \\ 225 & -60 & 16 & -4 & 1 & 0 \end{bmatrix}.$$

In a second case, this layer is given by the $n \times n$ grid $\boldsymbol{\ell}_2$, with

$$\begin{aligned} \boldsymbol{\ell}_2(i,j) &= s_3(i-j-1), \quad 1 < j < n-2, \quad j+2 < i < n \\ \boldsymbol{\ell}_2(i,j) &= s_3(j-i-1), \quad 3 < j < n, \quad 1 < i < j-2. \end{aligned}$$

For $n = 6$,

$$\boldsymbol{\ell}_2 = \begin{bmatrix} 0 & 0 & 1 & -4 & 16 & -60 \\ 0 & 0 & 0 & 1 & -4 & 16 \\ 1 & 0 & 0 & 0 & 1 & -4 \\ -4 & 1 & 0 & 0 & 0 & 1 \\ 16 & -4 & 1 & 0 & 0 & 0 \\ -60 & 16 & -4 & 1 & 0 & 0 \end{bmatrix}.$$

In both cases, this layer can be rotated and positioned as described below to create additional null vectors. These layers ℓ_1 and ℓ_2 are applied one at a time in one function at a time giving us the following null vectors,

$$\boldsymbol{v}_{5,p}(p,j,k) = \boldsymbol{\ell}_1(j,k),$$

for $1 \le p \le n$, $(1 \le j \le n, 1 \le k \le n)$,

$$\boldsymbol{v}_{5,p}(p-n,j,k) = \boldsymbol{\ell}_2(j,k),$$

for $n+1 \le p \le 2n$, $(1 \le j \le n, 1 \le k \le n)$,

$$\boldsymbol{w}_{6,p}(i,p,k) = \boldsymbol{\ell}_1(i,k),$$

for $1 \le p \le n$, $(1 \le i \le n, 1 \le k \le n)$,

$$\boldsymbol{w}_{6,p}(i, p-n, k) = \boldsymbol{\ell}_2(i, k),$$

for $n+1 \le p \le 2n$, $(1 \le i \le n, 1 \le k \le n)$,

$$\boldsymbol{z}_{7,p}(i, j, p) = \boldsymbol{\ell}_1(i, j),$$

for $1 \le p \le n$, $(1 \le i \le n, 1 \le j \le n)$, and

$$\boldsymbol{z}_{7,p}(i, j, p-n) = \boldsymbol{\ell}_2(i, j),$$

for $n+1 \le p \le 2n$, $(1 \le i \le n, 1 \le j \le n)$. Note that the subscript changes between pairs above, indicating three separate sets of null vectors of this type. If we add the null vectors formed by rotated layers, we also have $\boldsymbol{u}_{*,p}$ for $2n+1 \le p \le 4n$, where $* = 5$, 6, or 7.

At this point we have defined n^3 null vectors of the first type, $6n^2$ null vectors of the second type, and $12n$ null vectors of the third type, giving a total of $n^3 + 6n^2 + 12n$ null vectors. As explained earlier, the matrix has dimension $n^3 + 6n^2 + 12n - 40$ in practice, so we have more null vectors than we can use. With some careful selection, however, we can create a set of linearly independent null vectors. Rather than explaining which vectors to use, we provide a list of vectors to omit from the complete set already described, namely $u_{4,*}$ where $* = 2n^2$, $2n^2-1$, $2n^2-2$, $2n^2-3$, $2n^2-2n$, $2n^2-2n-1$, $2n^2-2n-2$, $2n^2-2n-3$ (8 vectors), $u_{5,*}$ where $* = n-1$, n $2n-1$, $2n$, $3n-2$ $3n-1$, $3n$, $4n-2$, $4n-1$, $4n$ (10 vectors), $u_{6,*}$ where $* = n-1$, n $2n-1$, $2n$, $3n-3$, $3n-2$ $3n-1$, $3n$, $4n-3$, $4n-2$, $4n-1$, $4n$ (12 vectors), $u_{7,*}$ where $* = n-1$, n $2n-1$, $2n$, $3n-2$ $3n-1$, $3n$, $4n-2$, $4n-1$, $4n$ (10 vectors). Although we have not done a formal proof, empirically we observed that the remaining set of vectors is a null basis for the discrete curl operator for increasing n. The conditioning of the basis degrades as n increases, however, and we address that issue in Section 6.4.

6.3 Computational Results

In this chapter we have presented several options for finding a null basis, including

1. compute an orthogonal basis using an SVD factorization of the matrix (*svd*)
2. compute an orthogonal basis using a QR factorization of the transpose of the matrix (*qr*)
3. compute a sparse basis using a turnback method (or some other sparse null basis solver) (*turnback*)

4. compute a sparse basis using a turnback method, then orthogonalize the basis (*turnback qr*)

5. create a basis using explicit formulas (for specific operators and domains) (*explicit*)

6. create a basis using explicit formulas, then orthogonalize the basis (*explicit qr*).

In this section we compare these methods in terms of four characteristics: their sparsity, conditioning, computational complexity, and accuracy. The accuracy of a null basis relates to the question of whether we obtain the zero matrix when we premultiply the basis by our problem matrix. We measure accuracy using the *null residual*, which we define to be the maximum magnitude of any entry in the product of the original problem matrix and the null basis, or more simply, the max norm of that product. The goal, then, is a small null residual, since the smaller this value is, the closer all values in the product are to zero. Our graphs provide data for the absolute null residual as described above, though if desired, a relative null residual could be defined as the max norm of the product of the two matrices divided by the norm of each. Since the null residuals we compare to one another all use the same problem matrix and since most of the null bases are orthogonal, comparing the relative null residuals instead of absolute null residuals for our examples would change the numeric values on our graphs, but it would make very little difference in the location of the plots with respect to one another.

The choice of method for a particular application depends on the relative importance of the four characteristics listed above. It may be worth additional computing time for better sparsity. Or better conditioning may be worth tolerating more nonzero entries. For our primary application problem, the curl problem in three dimensions, the null space component composes at least one third of the complete problem matrix (see Appendix A.1). The way the problem is formulated (see Section 8.2), the same operator is used with several right-hand sides as the solution is computed through time. Therefore, we have at least two good reasons to ensure that our null basis has good sparsity, conditioning, and accuracy, with relatively less regard to the computational time.

First, we consider the computational complexity of these options. Throughout these calculations, we assume QR factorizations are computed using Householder transformations. Let $\boldsymbol{D}$ be our problem matrix of size $m \times p$, with $m < p$. For method 1, the number of operations is on the order of $pm^2 + m^3$ for a dense matrix [29]. Using ARPACK and computing the SVD as a sparse matrix may reduce the number of operations, though that routine is intended to compute only a handful of singular values and vectors. To compute a QR factorization of $\boldsymbol{D}^T$ (method 2), we need on the order of $pm^2 - m^3/3$ additions and multiplications for a dense matrix [29], possibly fewer for a sparse implementation.

method	number of operations
1. SVD	$\mathcal{O}(pm^2 + m^3)$
2. QR	$\mathcal{O}(pm^2 - m^3/3)$
3. turnback	$\mathcal{O}(mp^2 - 1/3(m^3 + p^{5/2} - mp^{3/2}))$
4. turnback and QR	$\mathcal{O}(2p^3 - p^{5/2} + mp^{3/2})$
5. explicit	$\mathcal{O}(p)$
6. explicit and QR	$\mathcal{O}(2p^3 - 3p^2m + m^3)$

Table 6.1: Number of operations to compute null basis

In the case of numeric row dependencies in the original matrix in addition to column dependencies due to the matrix structure, we have used two QR factorizations to compute the null basis. The first pass identifies dependent rows, and the second computes the QR factorization of a set of independent rows. This scenario occurs with the 3-D curl problem. It is not clear whether sparse implementations of these two methods produce sparser null bases; we used dense implementations to obtain the results in this work.

The number of operations for the turnback method (method 3) is unclear, as it depends on the number of nonzeros in each null vector. If the number of nonzeros per null vector is proportional to $\sqrt{p}$ and the number of null vectors is $p - m$, we estimate the number of operations to be about $mp^2 - 1/3(m^3 + p^{5/2} - mp^{3/2})$ (see Appendix A.2). For method 4, we add a QR factorization of the $p \times (p - m)$ basis to that, giving a total of about $mp^2 - 1/3(m^3 + p^{5/2} - mp^{3/2}) + p(p - m)^2 - (p - m)^3/3 = \mathcal{O}(2p^3 - p^{5/2} + mp^{3/2})$.

Explicit null basis formulas vary in number of operations, but in the worst case, creating an explicit basis is $\mathcal{O}(p)$, where p is the larger dimension of the matrix and corresponds to the total number of grid points. This result holds even in cases where a sequence of numbers generates the values of the null vector. Thus, method 5 is $\mathcal{O}(p)$ and method 6 is $\mathcal{O}(2p^3 - 3p^2m + m^3)$, the cost of the subsequent QR factorization.

For easy reference, the results above are summarized in Table 6.1. While it may appear the methods 4 and 6 are the most complex because of the large leading term, method 6 has a second term that cancels much of the first term. Method 4 may well take the most time to compute (the complexity is based on an approximate number of nonzeros per null vector, so we cannot be precise here), but the gain in sparsity over the SVD or QR method makes it preferable to them, since as mentioned earlier, sparsity is a more important factor in method choice than computational complexity for our primary application.

Now we turn to the other characteristics of the null bases. It is difficult to characterize these for a general matrix or problem, so we will consider several specific examples. For each example, we provide information for each of the methods above in terms of sparsity and accuracy. We provide information on conditioning only for nonorthogonal bases, produced by methods 3 and 5. In a few cases the turnback method implementation was unable to compute a basis

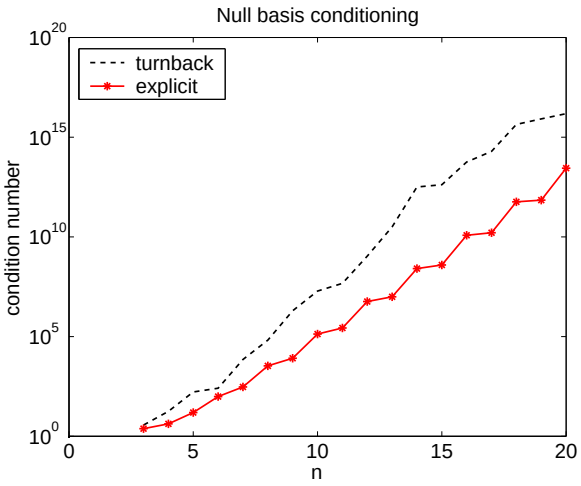

Figure 6.6: Condition numbers for finite difference 2-D Laplacian null bases

for finer discretizations. For those problems, we omit data points for methods 3 and 4 for the larger values of n.

As a first example, we consider the 2-D Laplacian operator discretized with finite differences. Figures 6.6-6.8 display data for conditioning, sparsity, and accuracy, respectively, as n, the number of mesh points in each dimension, increases. For both the turnback and explicit methods, the data are based on a scaled basis, with each null vector scaled by its maximum norm. As shown, the condition numbers for the nonorthogonal methods increase at an intolerable rate. A realistic simulation would likely use on the order of 50 data points per dimension, and at $n = 20$, both of these methods already have condition numbers over 10^{12}. Consequently, bases produced by both the turnback and explicit methods must be orthogonalized or their conditioning otherwise improved in order to be useful. As the condition numbers grow, however, direct orthogonalization techniques incur greater difficulty in retaining the accuracy of the bases, as the larger null residuals indicate. An alternative to complete orthogonalization for improved conditioning in this case is discussed in Section 6.4. The explicit method far exceeds the other methods in regard to sparsity, accuracy, and computational complexity, and the turnback method is a good alternative to orthogonal methods as long as the conditioning can be controlled. Other specific problems confirm these trends.

Figures 6.9-6.11 display data for the conditioning, sparsity, and accuracy for the 2-D curl problem, discretized with finite differences. Here, the turnback basis is slightly sparser than our explicit basis, but its conditioning and accuracy are somewhat worse than those for the explicit basis. The turnback method additionally incurs more floating operations than the explicit method, leaving the explicit method the overall winner. The null residuals for the explicit method are all zero to machine precision, so no line shows up on the logarithmic plot.

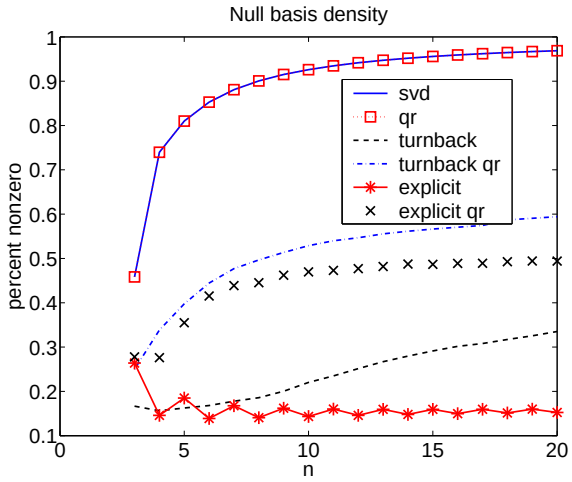

Figure 6.7: Density of finite difference 2-D Laplacian null bases

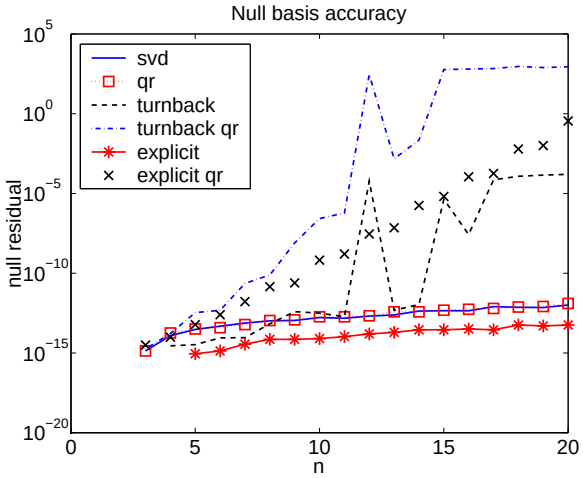

Figure 6.8: Accuracy of finite difference 2-D Laplacian null bases

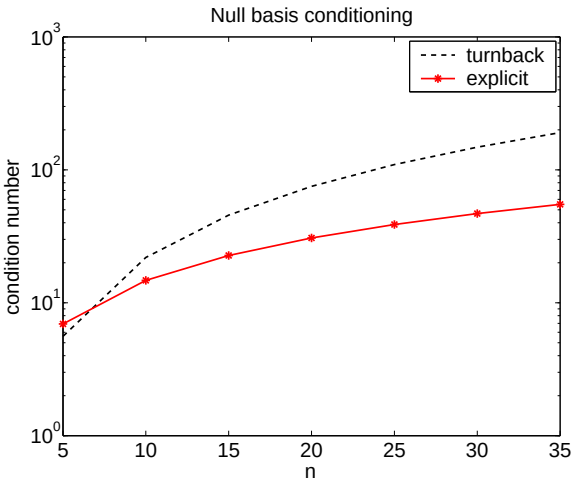

Figure 6.9: Condition numbers for finite difference 2-D curl null bases

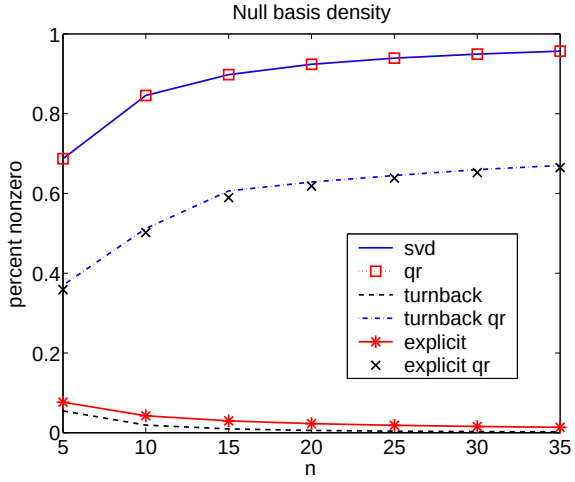

Figure 6.10: Density of finite difference 2-D curl null bases

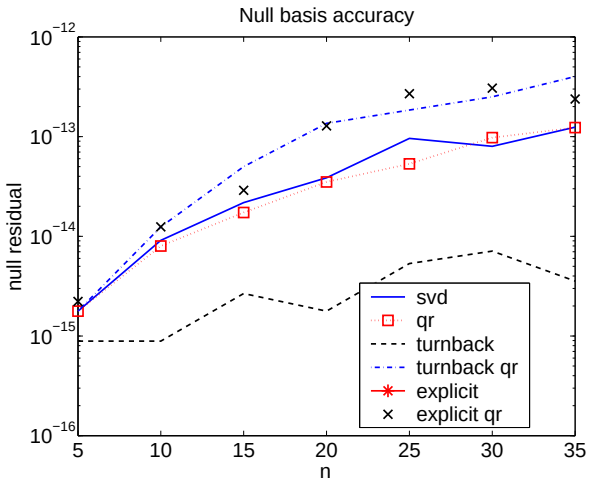

Figure 6.11: Accuracy of finite difference 2-D curl null bases

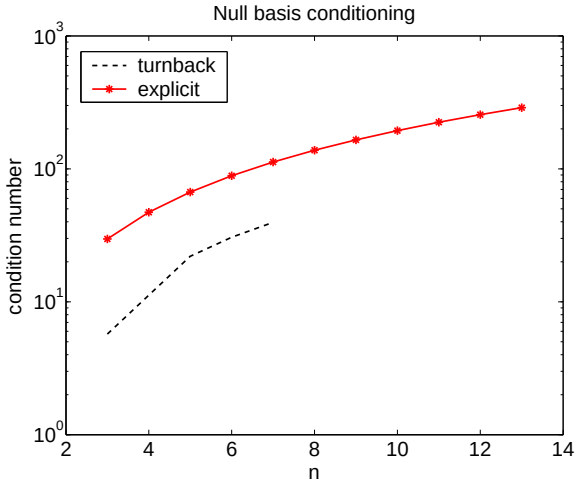

Figure 6.12: Condition numbers for finite difference 3-D curl null bases

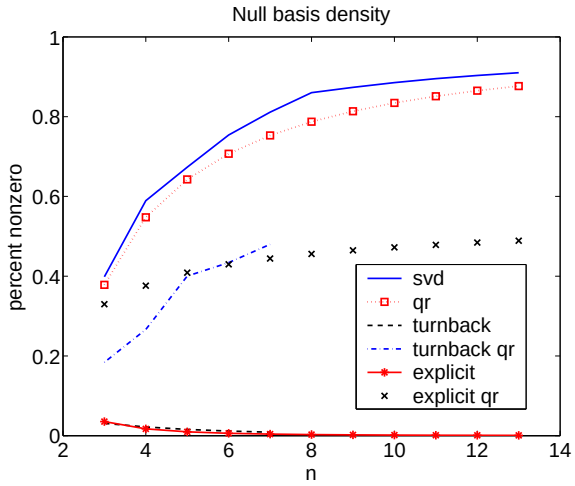

Figure 6.13: Density of finite difference 3-D curl null bases

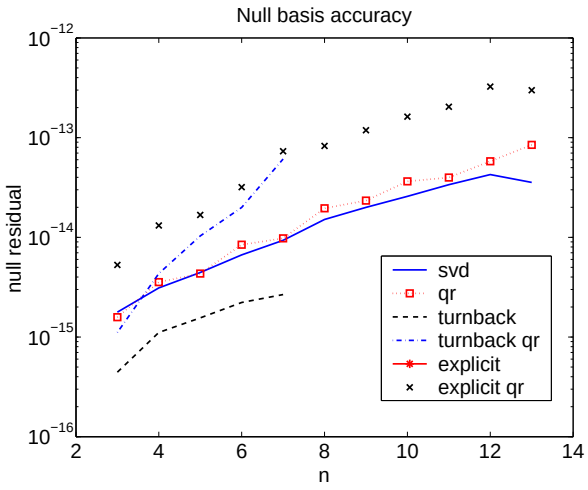

Figure 6.14: Accuracy of finite difference 3-D curl null bases

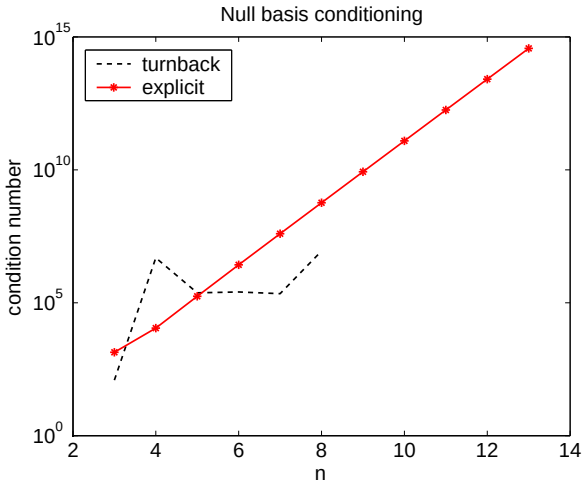

Figure 6.15: Condition numbers for finite element 3-D curl null bases

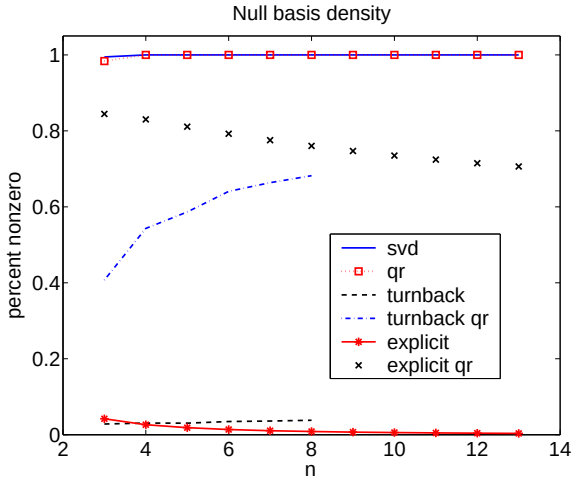

Figure 6.16: Density of finite element 3-D curl null bases

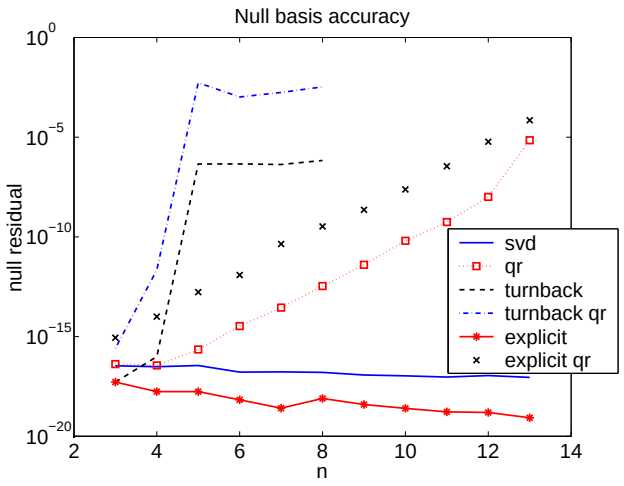

Figure 6.17: Accuracy of finite element 3-D curl null bases

Moving on to three-dimensional examples, Figures 6.12-6.14 display data for the conditioning, sparsity, and accuracy for the 3-D curl problem, discretized with finite differences, and Figures 6.15-6.17 display data for the same problem discretized with square, bilinear finite elements. There are two special notes about these results. First, the turnback method did not give a complete basis for the finite element discretization. The limited data here are gleaned from the null vectors provided. The condition numbers and density values may be a bit misleading, however, as additional null vectors may not be as independent or as sparse as those given. Second, for these problems, we used two QR factorizations rather than just one for method 2 to deal with dependent rows in the original matrix. Using MATLAB, this is still notably faster than computing the SVD of the matrix for larger values of n. We purposely omit computing times for most of these calculations due to optimizations in MATLAB and non-optimized implementations of some of these algorithms. As one example, however, for the finite-difference discretization of the curl in 3-D with $n = 13$, the computing times using MATLAB on a Sun V880 750 MHz processor were approximately 4.6 hours, 33 minutes, and 4 seconds respectively for the SVD, QR, and explicit bases. For these problem sizes, the SVD becomes impractical to compute. As previously mentioned, the implementation of the turnback method available to us was unable to compute a null basis for this problem size. Since it appears from the density values given for small values of n that the number of nonzeros per null vector is less than the square root of the number of matrix columns, we assume the computational complexity for turnback is better than what we estimated earlier and that the turnback method or one of the other methods mentioned in Section 6.1 would be viable for computing a sparse basis for a problem of this size.

For the finite-difference discretization, all the methods are roughly similar in conditioning and accuracy. The sparsity for both the turnback and explicit methods is excellent and about the same. Computationally, the explicit method wins over turnback, but given a proper implementation, the turnback method could be a good choice for finding the null basis in a case where an explicit method were not available.

Results for the finite-element discretization are a little more complicated, with poor results for all methods in one characteristic or another as the problem size grows. The SVD and QR factorizations are almost completely dense. The explicit method followed by a QR factorization is not much better, though it does improve as the problem size grows. All but the SVD and explicit methods have trouble maintaining accuracy as the problem grows, and the explicit method suffers from poor conditioning as the problem grows. Fortunately, we can improve the conditioning of the explicit method somewhat without incurring too much fill, as explained in the following section. The turnback method, after showing positive results for all previous examples, struggles here. The reason for its inconsistent performance for this problem is not understood and warrants additional research. It may be a simple matter of tweaking some parameters in the implementation, but it may be more complicated, involving the details of this particular example.

In summary, these examples show trends of poor conditioning for the turnback and explicit methods for some problems, they show accuracy declining as the problem size grows for all methods, with methods 4 and 6 having the most difficulty, and they show the explicit and turnback methods dominating with respect to sparsity. The poor conditioning of explicit methods for particular problems is addressed in the next section, and the issue of declining accuracy of a null basis due to orthogonalization is further discussed in Appendix A.4.

6.4 Improving Conditioning

The heuristic algorithms described in Section 6.1 and the explicit bases provided in Section 6.2 focus on sparsity but not on the conditioning of the resulting basis. While sparsity and good conditioning can occur together, e.g., the identity matrix, one is often gained at the expense of the other. The option of computing an orthogonal basis, either from scratch, using an SVD or QR factorization, or from a known sparse basis was considered in several methods in Section 6.3. Those methods solve the conditioning problem at the expense of sparsity. Stewart [51] suggests that a QR factorization of a sparse matrix (such as in methods 4 and 6 of Section 6.3) does not require storage of a resulting dense matrix if approached in an intelligent way. Specifically, if a $p \times k$ matrix $\boldsymbol{Z}$ ($p > k$) is a sparse null basis, and $\boldsymbol{Z} = \boldsymbol{QR}$, with $\boldsymbol{Q}$ orthogonal and of dimension $p \times k$ and $\boldsymbol{R}$ of dimension $k \times k$, then $\boldsymbol{Q}$ is an orthogonal basis for the null space. Rather than storing the likely dense matrix $\boldsymbol{Q}$ directly, however, we can store only $\boldsymbol{Z}$ and $\boldsymbol{R}$.

The null basis must have full rank by definition, and since $\boldsymbol{Q}$ is orthogonal, $\boldsymbol{R}$ is nonsingular. It follows that $\boldsymbol{Q} = \boldsymbol{Z}\boldsymbol{R}^{-1}$, with $\boldsymbol{Z}$ sparse, and $\boldsymbol{R}$ likely sparse and of smaller dimension than $\boldsymbol{Q}$. Thus, we can store and use our orthogonal basis $\boldsymbol{Q}$ as a product of two sparse matrices, not directly inverting $\boldsymbol{R}$, but simply solving an upper triangular system of equations whenever needed.

Stewart's paradigm can be useful, and in some cases it may be the best option for a sparse, well-conditioned matrix, but it requires the storage of two matrices and may increase the null residual of the matrix (see accuracy graphs of of methods 4 and 6 in Section 6.3.) As an alternative, we developed a couple of heuristics to improve conditioning of a sparse matrix while maintaining as much sparsity as possible. The first algorithm is effective for for improving the conditioning of explicit bases for the 2-D Laplacian operator dicretized with finite differences. It experiences nominal density gain with appreciable improvement in the condition number. It is further explained in Section 6.4.1. The second algorithm works for improving the conditioning of explicit bases for the 3-D curl operator discretized with finite elements (when the null vectors are ordered appropriately) and shows promise for other matrices as well. This second heuristic is discussed in Section 6.4.2. Both heuristics are applicable both to square and overdetermined matrices.

6.4.1 Column Pair Orthogonalization

One heuristic for improving conditioning is the column pair orthogonalization heuristic. The motivation behind the heuristic is the fact that ill-conditioning occurs when one column of a matrix is nearly dependent on another column or on a set of columns. If the first is true, say column $\boldsymbol{a}$ is nearly dependent on some other column $\boldsymbol{b}$, then the two columns must have similar sparsity patterns. Following this reasoning, the condition improver searches for columns with similar sparsity patterns and orthogonalizes them against each other. The fill caused by orthogonalizing column $\boldsymbol{a}$ against column $\boldsymbol{b}$ occurs where $\boldsymbol{a}$ has nonzeros that $\boldsymbol{b}$ does not have. (By orthogonalizing column $\boldsymbol{a}$ against column $\boldsymbol{b}$, we mean setting $\boldsymbol{b} = \boldsymbol{b} - (\boldsymbol{a}^T\boldsymbol{b}/\boldsymbol{a}^T\boldsymbol{a})\boldsymbol{a}$.) If the sparsity patterns of $\boldsymbol{a}$ and $\boldsymbol{b}$ are similar, the resulting fill will be minimal.

```
ColumnPairOrth (numPerRound, numRounds)
     loop numRounds times
1.   search for column pairs with similar nonzero patterns
     a. for each pair of columns
          - record number of nonzeros in common
          - record maximum fill if orthogonalized against each other
     b. sort by incr. maximum fill then by decr. common nonzero values
2.   orthogonalize numPerRound pairs of columns based on sorted data
```

This algorithm is effective for improving the conditioning of the explicit basis explained in Section 6.2.1. As an example, Figure 6.18 shows results for

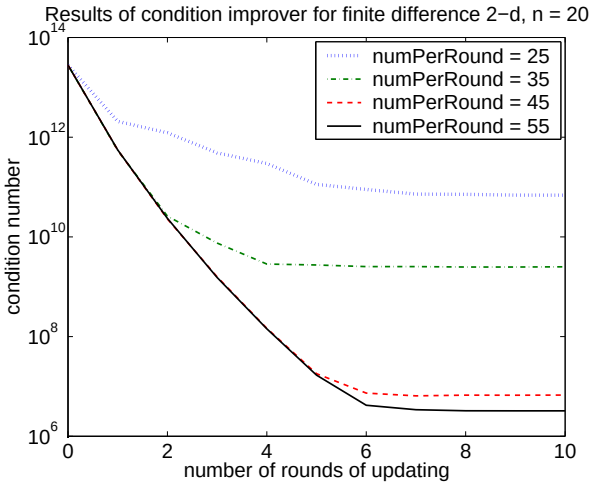

Figure 6.18: Results of condition number improver for 2-D Laplacian

ten iterations of updating for the null basis matrix for problem size $n = 20$. The value at iteration 0 is the condition number for the matrix with each column scaled by its maximum norm. The various lines represent different values for the number of orthogonalizations performed per iteration. This is a heuristic algorithm. We do not yet have a specific way to choose the optimal values for number of rounds of updating or number of pairs to orthogonalize per round.

To understand the effectiveness of the algorithm, it is important to consider not only the resulting condition number after the update (up to seven orders of magnitude in this example!) but also the density and null residuals of the resulting matrices. The density of the matrices after ten iterations with the specified value for numPerRound are shown in Figure 6.19, where we see that the density gain is very small compared to the improvement of the condition number. The null residuals for these resulting matrices, shown in Figure 6.20, could be better but are tolerable. This heuristic provides a good alternative to completely orthogonalizing the 2-D Laplacian explicit null basis. This step obviously adds to the total computational time, and the null residuals are not ideal, but they are better than the null residual of the orthogonalized explicit basis, and the heuristic improves the condition number considerably while minimally increasing the density.

6.4.2 Threshold QR Factorization

The algorithm of the previous section works well for the 2-D finite-difference Laplacian explicit null basis, but it fares much less well in improving the condition number of the 3-D finite-element curl explicit null basis. For this problem we found that a partial QR factorization is the key to improving conditioning without suffering fill comparable to the density of a complete orthogonalization.

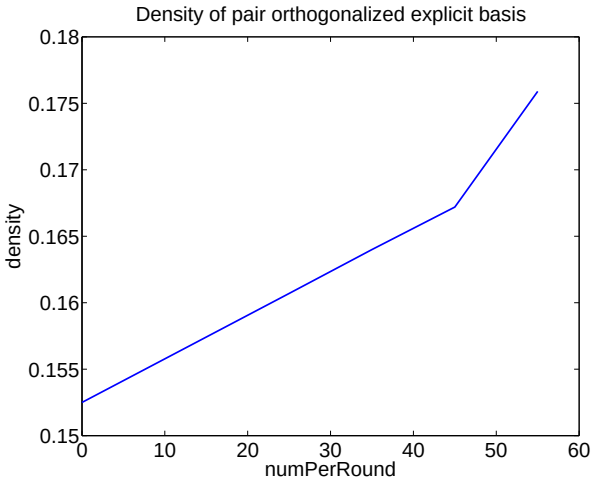

Figure 6.19: Results of condition number improver for 2-D Laplacian: sparsity

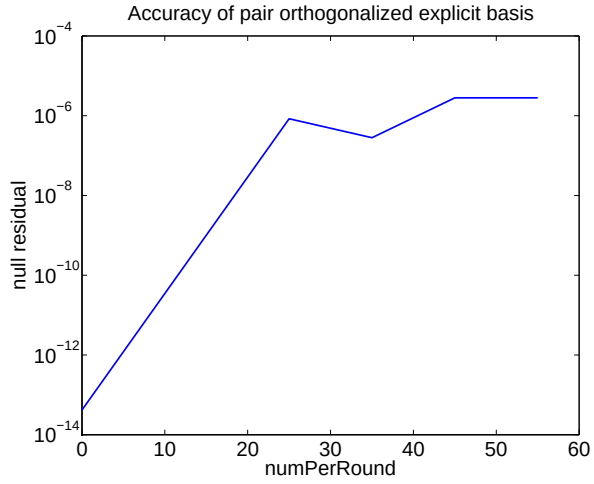

Figure 6.20: Results of condition number improver for 2-D Laplacian: accuracy

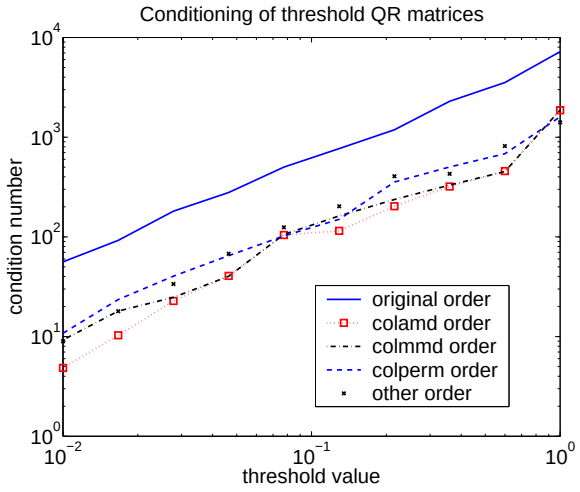

Figure 6.21: Threshold QR results for 3-D curl explicit basis ($n = 5$)

The method is based on modified Gram-Schmidt orthogonalization, in turn normalizing each column and orthogonalizing it against all subsequent columns, as long as the inner product of the normalized vector and the column to be updated is greater than a specified tolerance.

```
ThresholdQR (inMat, tol)
i = 1 to numColumns of inMat
    * normalize column i
    ( orthogonalize column i against remaining columns )
    * j = i + 1 to numColumns of inMat
        - if ⟨i, j⟩ ≥ tol
              orthogonalize column i against column j
        - end if
    * end loop j
end loop i
```

As in an LU factorization or complete QR factorization, the amount of fill generated is sensitive to the column ordering of the matrix. For our 3-D finite-element basis, any of the MATLAB built-in reordering functions `colamd` (column approximate minimum degree ordering), `colmmd` (column minimum degree ordering), and `colperm` (column permutation based on increasing nonzero count) decreased the fill (and thus improved the sparsity of our updated basis) immensely compared to the original ordering of the explicit basis. Figures 6.21 and 6.22 show the conditioning and density of our explicit basis for various threshold values with problem size $n = 5$. Also included in the plots is an ordering of the explicit basis with the second type of null vectors placed before those of the first type. As this ordering produces similar results to the non-specific `colamd` and `colmmd` orderings, we did not pursue a smarter ordering of null vectors by hand. The original condition number of the basis is 3.40×10^5, and the original density is

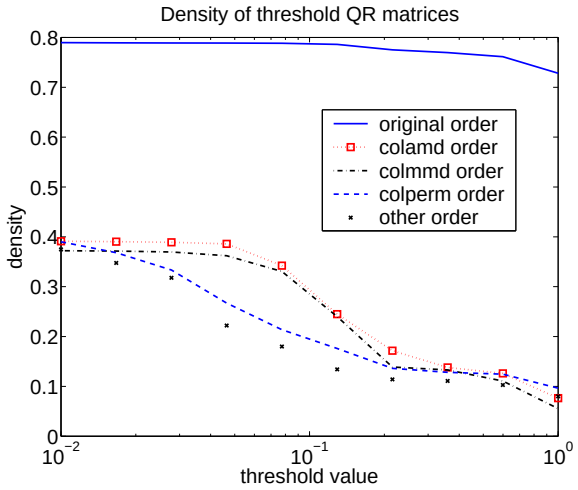

Figure 6.22: Threshold QR results for 3-D curl explicit basis ($n = 5$)

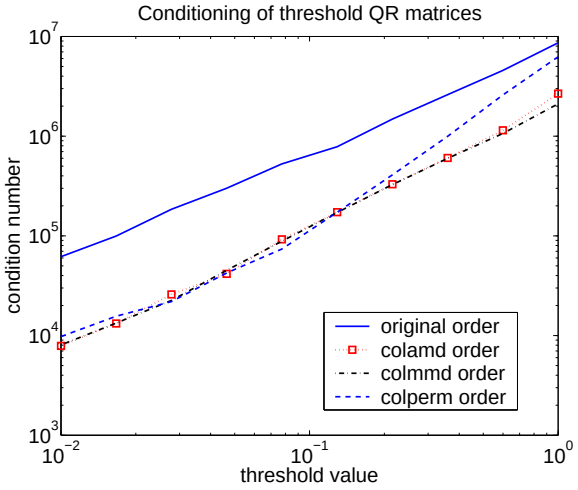

Figure 6.23: Threshold QR results for 3-D curl explicit basis ($n = 10$)

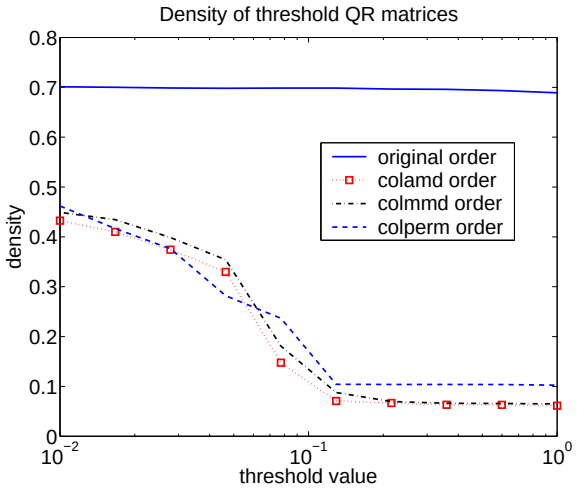

Figure 6.24: Threshold QR results for 3-D curl explicit basis ($n = 10$)

just under 2 percent. The graphs suggest a threshold value around 0.1 would improve the conditioning by 3 orders of magnitude while increasing the density to around 20 percent, depending on the ordering. This is a better option than the QR factorization (assuming sparsity is a goal), which improves the conditioning by 5 orders of magnitude but increases the density to over 45 percent. The results are similar for $n = 10$, as Figures 6.23-6.24 demonstrate.

To test this method on other matrices, we gleaned sparse matrices with poor conditioning from the Matrix Market hosted by NIST [8]. Figures 6.25-6.28 show the conditioning and density values for several threshold values and several initial column orderings of three sparse matrices with different sparsity structures. The first matrix, west0381, has an unusual, nonsymmetric sparsity pattern. The nonzeros form right angle patterns which are spread throughout the matrix. The condition number of the matrix is approximately 1.26×10^6 and its density is 0.0147 or 1.47 percent. The second matrix, nos7, has a strong diagonal pattern with a small band at the diagonal and one band each of subdiagonal and superdiagonal entries off the main diagonal. It's original density is 0.0087, and its condition number is 2.37×10^9. Our third example is the matrix fs_183_6 which appears to have a symmetric sparsity pattern. It has a banded diagonal, patterned super and subdiagonal swoops (not straight diagonals), and several nonzeros dotted elsewhere throughout the matrix. The density is 0.0299 and the condition number is 1.74×10^{11}.

Each of these examples shows the trade-off between conditioning and sparsity. In all cases, it is possible to reduce the magnitude of the condition number to about the square root of the original condition number while maintaining a density of less than 20 percent, given the correct threshold value and initial matrix ordering. In the case of the fs_183_6 matrix, this heuristic is superb, giving a decrease in the condition number of 9 orders of magnitude while merely

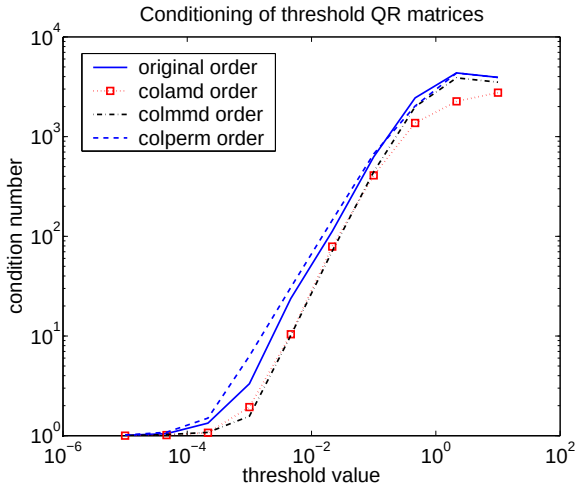

Figure 6.25: Threshold QR results for west0381 matrix

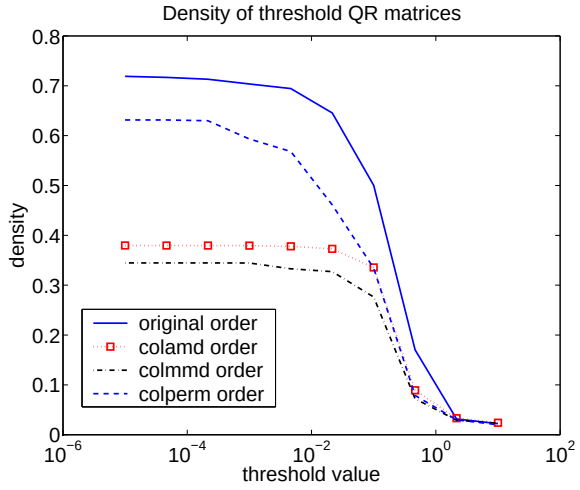

Figure 6.26: Threshold QR results for west0381 matrix

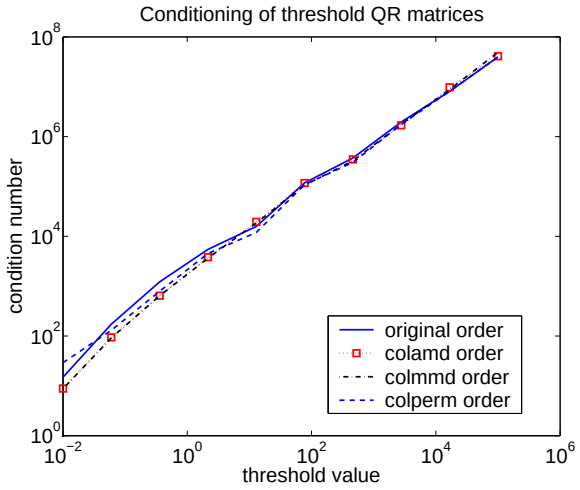

Figure 6.27: Threshold QR results for nos7 matrix

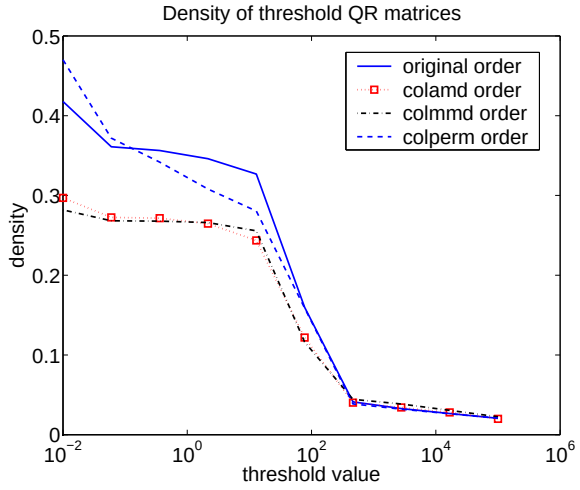

Figure 6.28: Threshold QR results for nos7 matrix

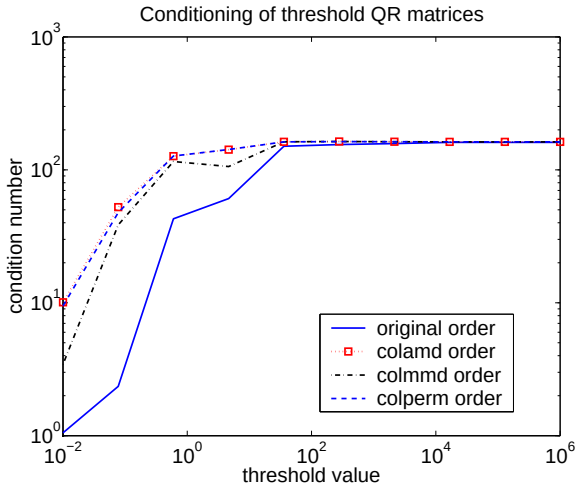

Figure 6.29: Threshold QR results for fs_183_6 matrix

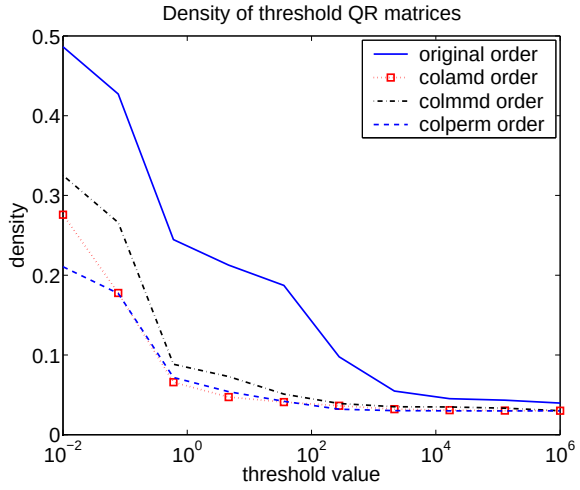

Figure 6.30: Threshold QR results for fs_183_6 matrix

doubling the original density. (This can be seen on the graphs at the threshold value close to 10^6.) This condition improver does not automatically choose a threshold value–it requires user involvement, but it is a useful tool for improving the condition number of a matrix while incurring a "reasonable" amount of fill. Thus it provides a good alternative to a complete orthogonalization of the explicit null basis for the 3-D finite element curl.

7 Solving the Linear System

After discretizing a differential equation and obtaining a discrete null basis, we put the pieces together as described in Chapter 5 to create the linear system with full column rank corresponding to our discetized approximation. Solving the resulting system is the final step in obtaining a solution using the null-space approach. Preferably, we would solve the system both quickly and accurately, but sometimes those goals oppose each other. This chapter presents and discusses several options for solving the system, which we denote

$$\boldsymbol{C}\boldsymbol{x} = \boldsymbol{d}$$

in its complete form, with $\boldsymbol{C}$ an $m' \times p$ matrix with $m' \geq p$, $\boldsymbol{d}$ an $m' \times 1$ vector, and $\boldsymbol{x}$ a $p \times 1$ vector, and as

$$\begin{bmatrix} \boldsymbol{D} \\ \hat{\boldsymbol{Z}}^T \end{bmatrix} \boldsymbol{x} = \begin{bmatrix} \boldsymbol{a} \\ \boldsymbol{b} \end{bmatrix}, \tag{7.1}$$

in its original partitioned form (see (5.1)), with $\boldsymbol{D}$ an $m \times p$ matrix with $m \leq p$, $\boldsymbol{a}$ an $m \times 1$ vector, $\hat{\boldsymbol{Z}}$ a $p \times k$ matrix with $k = p - \text{rank}(\boldsymbol{D})$, $\boldsymbol{b}$ a $k \times 1$ vector, and our unknown $\boldsymbol{x}$ a $p \times 1$ vector.

7.1 Problem Characteristics

If our matrix $\boldsymbol{C}$ were well-conditioned and had "nice" characteristics such as symmetry and sparsity, this chapter would be unnecessary because several existing linear solvers, direct and iterative, could easily solve the problem. The formulation of our problem, however, produces a system that is not simple to solve.

The original problem matrix $\boldsymbol{D}$ will be sparse for finite-difference discretizations and for finite-element discretizations with localized elements. As Chapter 6 details, we can generally find a sparse null basis for our original problem matrix, though some of that gain is less evident in the finite element case because we premultiply our null basis when we create the null basis portion of the complete system ($\hat{\boldsymbol{Z}}^T$). Even so, our system does have the positive characteristic of sparsity, which is useful if we can exploit the zeros as we compute a solution. The sparsity of the complete matrix $\boldsymbol{C}$ is dependent on the sparsity of $\hat{\boldsymbol{Z}}^T$ because, while the null basis constraints compose less and less of the system as the prob-

lem size grows, they provide at least one-third of the total constraints in our motivating example (see Appendix A.1).

We do not find symmetry in $\boldsymbol{C}$ even with reordering the matrix. Although discretized differential equations generally benefit from symmetry, the process of omitting the boundary-related equations and adding null-space constraints removes that advantage in the null-space approach.

In some cases, we also face poor conditioning of $\boldsymbol{C}$ as the problem size grows. The poor conditioning is most problematic for the finite difference discretization of the Laplacian operator used in combination with its explicit null basis.

7.2 Direct Methods

Direct methods for solving linear systems are one option for solving our system. Various re-orderings of the matrix, such as nested dissection, minimum degree, and band orderings exploit sparsity in the matrix, and direct methods are generally robust even for ill-conditioned problems. Direct methods are also straightforward to implement and do not require preconditioning. (See [17, 19, 23] for more information on sparse direct methods.) With the memory and computing power of machines available today, direct methods can be used to solve moderately large problems, and in fact, we use direct methods to solve the application problems we present in Chapter 8. For very large problems, however, solving the system directly may be infeasible even with a powerful computer; in that situation, we must turn to iterative methods.

7.3 Iterative Methods

Iterative methods can easily take advantage of sparsity and can be very cheap computationally compared to direct methods. Iterative methods are most useful when the eigenvalues of a matrix are clustered, resulting in quick convergence to the solution and few required iterations. This advantage becomes more evident as the problem grows. Since our matrix $\boldsymbol{C}$ is sparse and will be large for a fine discretization of a differential equation, iterative methods seem to be a good option.

As a first example, we apply the general GMRES method (symmetric iterative methods do not apply as the characteristics given above imply) to the 2-D Laplacian on the unit square, discretized with finite differences, with an orthogonal null basis computed using MATLAB's `null()` routine. Discretized evenly with n nodes in each direction, the problem produces an $n^2 \times n^2$ system. The GMRES method [45], applied to the system with no preconditioner and with an error tolerance of 10^{-6}, requires on the order of n^2 iterations to converge, which is a very poor convergence rate. The convergence curves for $n = 5$, 10, 15, 20, and 25, using the MATLAB built-in GMRES solver, are shown in Figure (7.1).

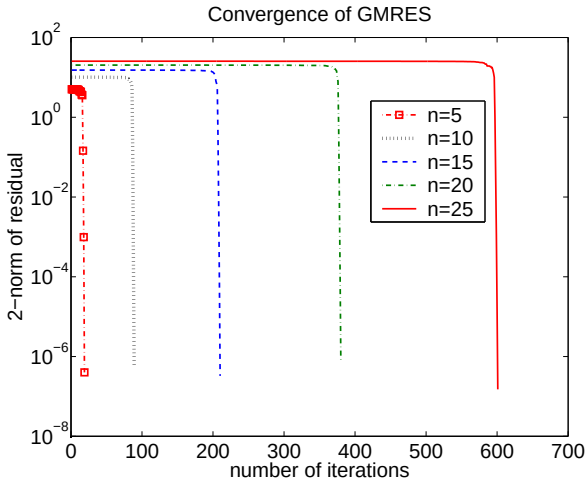

Figure 7.1: Convergence curves for GMRES with no preconditioner

The initial guess is a vector of ones. The graph confirms poor convergence rates for the GMRES method applied to this problem, with little convergence until the number of iterations almost equals the number of columns in the matrix. A look at the eigenvalues explains the difficulty GMRES has in finding the solution. Figure 7.2 plots the eigenvalues of $\boldsymbol{C}$ for $n = 15$. We used an orthogonal basis here to demonstrate the poor eigenvalue spread and convergence rates without additional conditioning problems due to calculation of the null space. In fact, the condition number of $\boldsymbol{D}$ is about 49, and the condition number of $\hat{\boldsymbol{Z}}$ is 1, but the condition number of the combined matrix $\boldsymbol{C}$ is about 1550. Even though the conditioning of the problem is reasonable, a preconditioner will be necessary to make GMRES viable.

7.4 Preconditioned Iterative Methods

Iterative methods converge speedily when the eigenvalues of the system are clustered. Since multiplying by a nonsingular matrix does not change the solution of a linear system, preconditioners (created as nonsingular matrices) are applied to the system in an attempt to improve the conditioning and eigenvalue clustering of the system matrix. Ideally, a preconditioner would maintain sparsity and be easy to invert (i.e., easy to solve systems with). Since linear systems vary considerably, preconditioners tend to be specific to certain types of problems. Many of the preconditioners we considered, such as banded preconditioners, polynomial preconditioners (described in [13]), and SPAI preconditioners (discussed in [16, 26] among others), either do not apply to our problem or provide little to no help in clustering eigenvalues enough to assist the convergence of iterative methods. Incomplete LU (ILU) factorization based on thresholding, however [38], provides us with preconditioners that are generally effective for the various

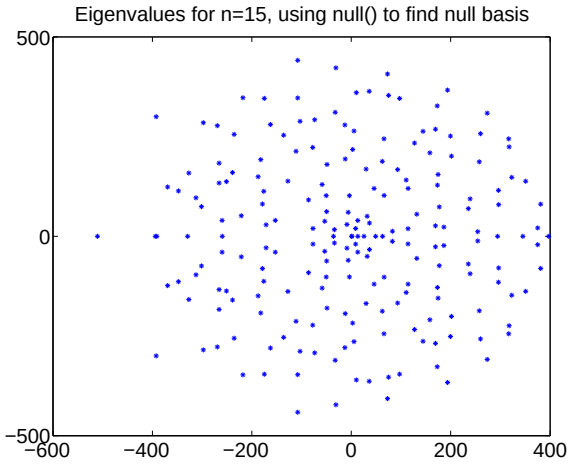

Figure 7.2: Eigenvalues of complete system for $n = 15$

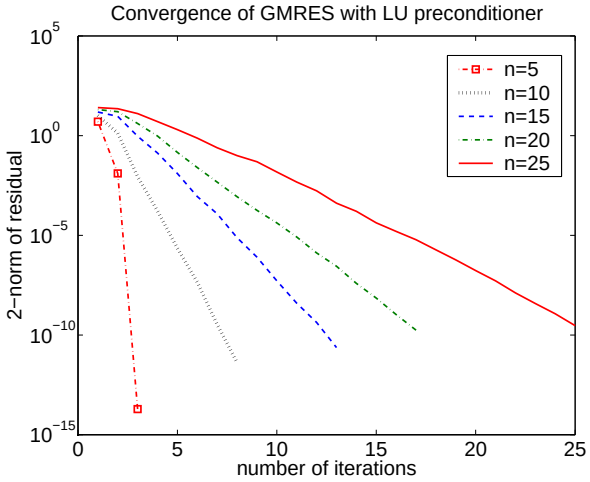

Figure 7.3: Convergence curves for GMRES with ILU preconditioner

operators and discretizations we considered. As a first example, compare the convergence curves for the 2-D Laplacian using ILU preconditioners (plotted in Figure 7.3) to the original convergence curves with no preconditioner (plotted in Figure 7.1). The ILU preconditioners for this problem had a drop tolerance of 10^{-4}.

Incomplete LU factorization works as a forward preconditioner. That is, we find a matrix $\boldsymbol{M}$ that approximates our system matrix $\boldsymbol{C}$ and solve the system $\boldsymbol{M}^{-1}\boldsymbol{C}\boldsymbol{x} = \boldsymbol{M}^{-1}\boldsymbol{d}$. The method of incomplete LU factorization based on thresholding finds an approximate LU factorization of $\boldsymbol{C}$ by applying the usual factorization method but dropping values less than a certain tolerance. In this way, the factorization takes less time than a complete factorization, and the $\hat{\boldsymbol{L}}$ and $\hat{\boldsymbol{U}}$ factors are sparser than the exact $\boldsymbol{L}$ and $\boldsymbol{U}$ factors. Since the approximate factors are lower and upper triangular, respectively, they are convenient for solving systems; they merely add a number of operations on the same order as matrix-vector multiplication. This method works well with iterative methods for our system, as the examples in Section 7.6 show, but it requires extra memory space as even approximate factors exhibit greater density than the original matrix.

7.5 Other Perspectives

In addition to considering various solution techniques for our linear system, we have also considered options for rewriting our problem. In this section, we present these additional ways to write the problem and discuss the benefits and liabilities of each.

7.5.1 Optimization Formulations

The form of our original partitioned problem, given in (7.1) above, is suggestive of an optimization problem with one set of equations deemed a minimization problem and the other set of equations deemed the constraints. First, we consider the upper set to be the minimization problem and the lower set the constraints. Since an exact solution to (7.1) will minimize the 2-norm of the residual $\boldsymbol{a} - \boldsymbol{D}\boldsymbol{x}$ and satisfy the constraints, $\hat{\boldsymbol{Z}}^T\boldsymbol{x} = \boldsymbol{b}$,

$$\min_x \quad \frac{1}{2}\|\boldsymbol{a} - \boldsymbol{D}\boldsymbol{x}\|_2^2 \quad \text{s.t. } \hat{\boldsymbol{Z}}^T\boldsymbol{x} - \boldsymbol{b} = 0,$$

is a valid way to rewrite the problem. When expanded and converted into a single system using Lagrange multipliers, the problem becomes

$$\begin{bmatrix} \boldsymbol{D}^T\boldsymbol{D} & \hat{\boldsymbol{Z}} \\ \hat{\boldsymbol{Z}}^T & \boldsymbol{0} \end{bmatrix} \begin{bmatrix} \boldsymbol{x} \\ \boldsymbol{y} \end{bmatrix} = \begin{bmatrix} \boldsymbol{D}^T\boldsymbol{a} \\ \boldsymbol{b} \end{bmatrix}, \tag{7.2}$$

where the components of $\boldsymbol{y}$ are the Lagrange multipliers. This provides a second form of the discrete problem represented by the linear system in (7.1).

Similarly, switching the equations in the minimization problem, we obtain

$$\begin{bmatrix} \hat{\boldsymbol{Z}}\hat{\boldsymbol{Z}}^T & \boldsymbol{D}^T \\ \boldsymbol{D} & \boldsymbol{0} \end{bmatrix} \begin{bmatrix} \boldsymbol{x} \\ \boldsymbol{y} \end{bmatrix} = \begin{bmatrix} \hat{\boldsymbol{Z}}\boldsymbol{b} \\ \boldsymbol{a} \end{bmatrix}. \tag{7.3}$$

Thus we have three equivalent forms of the discretized problem. Both of the alternate forms given here are full-rank, square systems. The square shape of the matrices is easily confirmed by considering the dimensions of the submatrices and the formulation of the complete system. The nonsingular claim is proved in Appendix A.3.

The advantages of these alternate formulations include better conditioning of the system to be solved (at least in the case of (7.2) compared to (7.1) for our sample problems) and symmetry. Unfortunately, we do not have positive definite systems, which would allow use of the conjugate gradient iterative method. Nor can we apply the block preconditioners described in [47] (or similar, symmetric forms) as we had hoped. The problem for such preconditioners is not that the system is indefinite but that the $1,1$ block submatrix is singular, and attempts to find a matrix close to the submatrix that is nonsingular or to reorder the entire matrix to create a nonsingular $1,1$ block have not been effective. Additionally, these linear systems are larger than the original system so fewer iterations may take more total compute time due to the increase of work at each iteration.

7.5.2 Decoupled Problems

Finite difference discretizations, for which $\hat{\boldsymbol{Z}} = \boldsymbol{Z}$, allow us to decouple our problem provided $\boldsymbol{D}$ has full row rank, as we now show. Let $\boldsymbol{D}$ be the discretized operator of size $m \times p$ with $m < p$ and $\text{rank}(\boldsymbol{D}) = m$, and let $\boldsymbol{Z}$ be a null basis for $\boldsymbol{D}$. By construction, $\boldsymbol{DZ} = 0$. Thus, the columns of $\boldsymbol{Z}$ are orthogonal to the rows of $\boldsymbol{D}$. Equivalently, $\text{span}(\boldsymbol{D}^T) \perp \text{span}(\boldsymbol{Z})$. In addition, $\text{span}(\boldsymbol{D}^T) \cup \text{span}(\boldsymbol{Z}) = \mathbb{R}^p$. It follows that the solution vector $\boldsymbol{x}$, which is in $\mathbb{R}^p$, can be written as $\boldsymbol{D}^T\boldsymbol{v} + \boldsymbol{Z}\boldsymbol{w}$, where $\boldsymbol{v}$ and $\boldsymbol{w}$ are unknown.

Substituting into the problem (form (7.1)), we obtain equations

$$\begin{aligned} \boldsymbol{D}(\boldsymbol{D}^T\boldsymbol{v} + \boldsymbol{Z}\boldsymbol{w}) &= \boldsymbol{a}, \\ \boldsymbol{Z}^T(\boldsymbol{D}^T\boldsymbol{v} + \boldsymbol{Z}\boldsymbol{w}) &= \boldsymbol{b}. \end{aligned}$$

These simplify to

$$\begin{aligned} \boldsymbol{D}\boldsymbol{D}^T\boldsymbol{v} &= \boldsymbol{a}, \\ \boldsymbol{Z}^T\boldsymbol{Z}\boldsymbol{w} &= \boldsymbol{b}, \end{aligned}$$

which can be solved independently for $\boldsymbol{v}$ and $\boldsymbol{w}$. This formulation breaks the

problem into two symmetric positive definite subproblems. (To prove that the subproblems are positive definite, apply Lemma 2 from Chapter 5 with $\boldsymbol{C} = \boldsymbol{I}$. Symmetry follows from the commutativity of the Euclidean inner product.) If $\boldsymbol{D}$ and $\boldsymbol{Z}$ exhibit good conditioning, this decoupling is an excellent solution technique. It enables use of the conjugate gradient method, and, in the case where the same differential equation is solved with different null-space constraints, it allows reuse of solution vector $\boldsymbol{v}$. The drawbacks of this formulation include applicability limited to problems with full row rank, loss of information in forming $\boldsymbol{D}\boldsymbol{D}^T$ and $\boldsymbol{Z}^T\boldsymbol{Z}$, and squaring of the condition numbers of both $\boldsymbol{D}$ and $\boldsymbol{Z}$.

If a finite element discretization is used with set of orthonormal basis functions to describe the solution, we again have $\hat{\boldsymbol{Z}} = \boldsymbol{Z}$, and the argument above holds for finding the solution vector $\boldsymbol{x}$ with two independent linear systems. Of course, in the finite element case, the solution vector contains coefficients of the approximate solution, which is given by $\boldsymbol{V}\boldsymbol{x}$, where $\boldsymbol{V}$ holds the set of basis functions as described in Chapter 2.

7.6 Comparison Study

Now that we have presented several options for formulating and solving the linear system, we address the question of which is best for our problem. As usual, the answer depends on the importance of various factors, such as time and memory constraints, but comparisons between options can help clarify the decision of which method to use. In this section, we compare results of applying an iterative method to various formulations of the system (described below) in terms of number of iterations required for convergence to an error tolerance of 10^{-10} and in terms of memory required for storing the problem matrix and any preconditioners. The number of iterations does not directly indicate computation time (e.g., consider the case of a direct solve), but it gives a more definite comparison than processor time, which varies depending on implementation and computer speed. For this reason, we omit the direct solve from our comparisons because it requires one iteration by definition as a direct solver. Our value for memory assumes the matrices are stored in a sparse format and reports the total number of nonzero values in the system matrix and any preconditioners. We do not calculate or report additional memory used by the iterative method for storing residual vectors.

According to Chen [13, p. 110], it is the choice of preconditioner rather than the choice of iterative method that makes the most difference in convergence rate. Accordingly, we will make comparisons using only one iterative method, namely GMRES [45], which applies to nonsymmetric matrices as necessary for our system. (Bi-CGSTAB [55], a variant of the conjugate gradient method that solves nonsymmetric systems, is another applicable iterative method, but as it fails to converge for some of our examples, we chose to provide data only for GMRES.) We use full GMRES to obtain our results (that is, no restarting), and

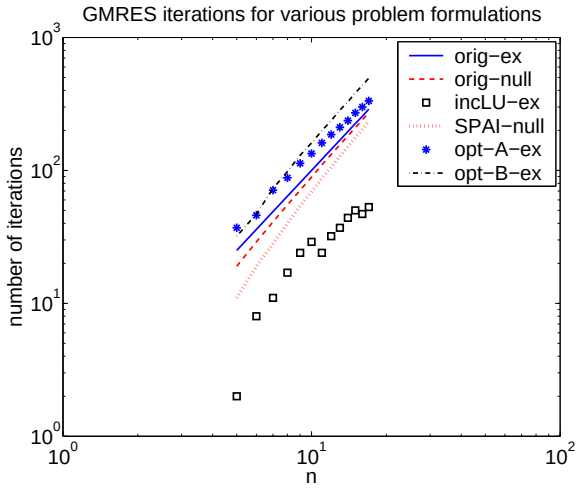

Figure 7.4: Convergence counts for GMRES applied to fin. diff. 2-D Poisson

we set the initial guess to be a random vector provided by MATLAB.

The forms of the system and various preconditioners we compare are given below. The brief identifying phrases listed in parentheses at the end of each description will be used to identify results on the plots, and the portion after the last hyphen indicates whether the rescaled explicit basis was used ("ex") or the orthogonal basis given by MATLAB's `null()` command ("null").

1. original system with no preconditioner using scaled explicit null basis (orig-ex)

2. original system with no preconditioner using orthogonal null basis (orig-null)

3. original system with incomplete LU factorization preconditioner found with appropriate tolerance (incLU-ex)

4. original system with SPAI preconditioner applied to rearranged linear system (SPAI-null)

5. reformulated system given in (7.2) (opt-A-ex)

6. reformulated system given in (7.3) (opt-B-ex).

For the SPAI preconditioner, we select the sparsity pattern of the original matrix as the sparsity pattern of the preconditioner, but we reorder the pattern to include more nonzero diagonal elements.

First, we return to the 2-D Poisson example, discretized with finite differences. The choice of α and β should not affect the convergence of the method when it is given a random initial guess, but we need to choose some problem to solve, and we set $\alpha = 12s^2t^2 + 2t^4$ and $\beta = s^2t^4$. The number of iterations for convergence of GMRES to a tolerance of 10^{-10} are shown in Figure 7.4.

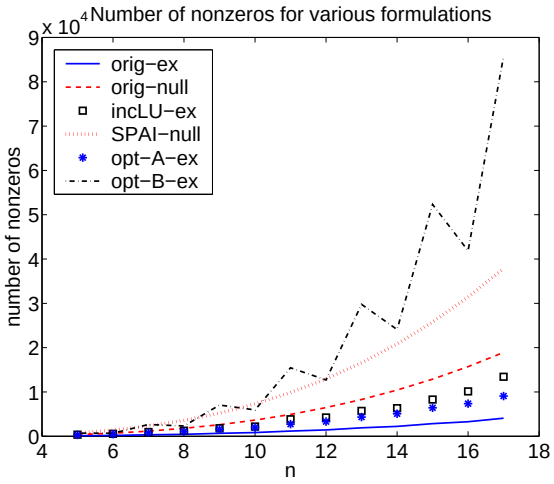

Figure 7.5: Number of nonzeros in matrices for finite difference 2-D Poisson

The explicit basis caused rank deficiency for $n = 18$, so we include data up to $n = 17$. The space requirements for the various systems are shown in Figure 7.5. The graphs plot the iteration counts and space requirements as the problem size grows.

We also include convergence and density results for the 3-D finite-element curl problem in Figures 7.6 and 7.7. The values for $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ have no particular significance here beyond giving us a specific problem to solve; they are

$$\boldsymbol{\alpha} = \begin{bmatrix} 12s^2t^2 + 2t^4 \\ 1 \\ 1 \end{bmatrix}, \qquad \text{and} \qquad \boldsymbol{\beta} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}.$$

Since GMRES solves square systems, we used a set of independent rows of the original matrix $\boldsymbol{D}$ in place of the complete matrix to obtain these results. (For a compatible system, any set of independent rows will do. The set of independent rows of $\boldsymbol{D}$ combined with rows created by the null-space constraints provides a square system.) Here the size of the reformulated systems incurred memory limitations for $n = 12$, so we include results for $n = 5$ to $n = 11$.

These two examples show similar trends. Both demonstrate that using the `null()` basis from MATLAB costs a lot in terms of memory while not buying much in terms of faster convergence. Similarly, the SPAI preconditioner does decrease the number of iterations for convergence of GMRES for a particular value of n, but the change in convergence values as the problem size grows mimics the change in convergence values of the original problem. Incomplete LU factorization as a preconditioner is the winner for convergence values for both examples. The drop tolerance was determined as the largest tolerance of the form 10^k for $k \in \{\mathbb{Z} | k < -2\}$ that gave nonsingular matrices $\boldsymbol{L}$ and $\boldsymbol{U}$. The convergence values shown on the graph dip where the drop tolerance was lowered

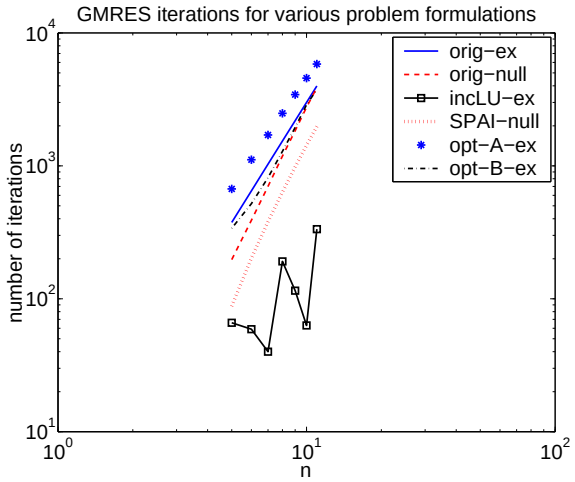

Figure 7.6: Convergence counts for GMRES applied to fin. el. 3-D curl

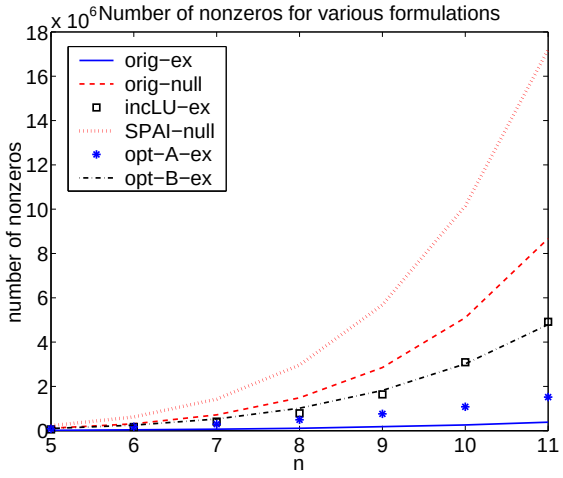

Figure 7.7: Number of nonzeros in matrices for finite element 3-D curl

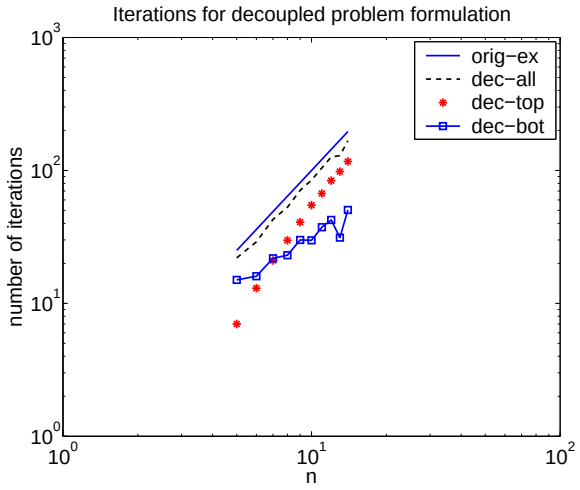

Figure 7.8: Convergence counts for CG applied to fin. diff. 2-D Poisson

by a factor of 10. For the 3-D curl problem, in Figure 7.6, we can connect the peaks of the graph to get an idea of the convergence trend as the problems size increases. The line is clearly less steep than the others. The $\boldsymbol{L}$ and $\boldsymbol{U}$ factors do require extra memory space, as the plots of memory requirements for the problems show, but all together, the problem does not take as much space as merely using a dense null basis in place of a sparse one.

Finally, for comparison purposes, we provide iteration counts for the conjugate gradient method applied to the decoupled formulation of the finite difference, 2-D Laplacian operator. The results (see Figure 7.8) are plotted in comparison to the number of GMRES iterations for the original formulation without a preconditioner. The explicit basis was used for both. The two parts of the decoupled problem are plotted separately ("dec-top" and "dec-bot" for the upper and lower portions respectively) and together ("dec-all"). The graph indicates that the number of iterations required to solve both parts of the decoupled problem is very similar to the number of iterations required to solve the original formulation of the problem using GMRES. This result may seem discouraging at first glance, but the iteration counts are improved just with reformulating the problem, and the symmetric positive definite feature of both portions of the decoupled problem gives much hope for the existence of effective preconditioners, perhaps even among the types that were not effective for our original problem formulation.

8 Applications

As explained in the introduction, the methods developed in this thesis were motivated by a problem in materials science. The null space approach applies to that problem but is useful in general for elliptic partial differential equations with either an infinite domain or a domain with unknown boundary conditions. In this chapter, we consider two specific applications, the curl problem that motivated this work in Section 8.2, and a Poisson problem, which is common in the field of electromagnetics.

Throughout this chapter we obtain results using sparse direct methods to solve the final linear system. The implementation is straightforward and the problems are small enough that we do not encounter memory limitations using a direct solver. In addition, solving the system directly removes a potential source of error, allowing us to observe how a finer grid affects the approximate solution without concern for possible effects of an iterative solver. When we refer to convergence in this chapter, then, we are considering the accuracy of the approximation compared to the exact solution as the grid is refined.

8.1 Poisson Equation in Electromagnetics

A common calculation in electromagnetics is that of finding the electrostatic potential of a field given a charge density over the field. The domain is in principle infinite because electrostatic charges do not abruptly lose effect at some point but gradually diminish in their effect as distance increases between two charges. Typically, the calculation is done on some finite domain of interest and the boundary values are estimated. Often the boundaries are extended beyond the region of interest to lessen the effect of boundary approximation in the pertinent part of the domain. Sometimes an approximation is found using a coarse discretization on a larger domain, and that approximate solution is used to formulate boundary conditions for the smaller domain of interest. The null space method provides an alternative to the imposition of such unknown or arbitrarily chosen boundary conditions and may allow us to find the desired solution using a smaller domain than a conventional method would require.

The problem is described mathematically using the Poisson equation

$$\Delta u = -f, \tag{8.1}$$

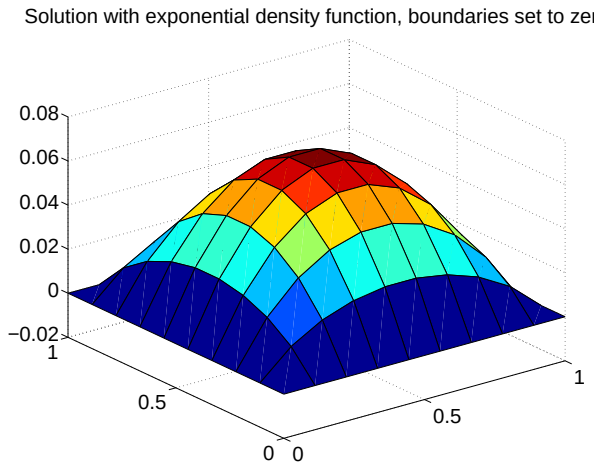

Figure 8.1: Specific solution to $\Delta u = -f$

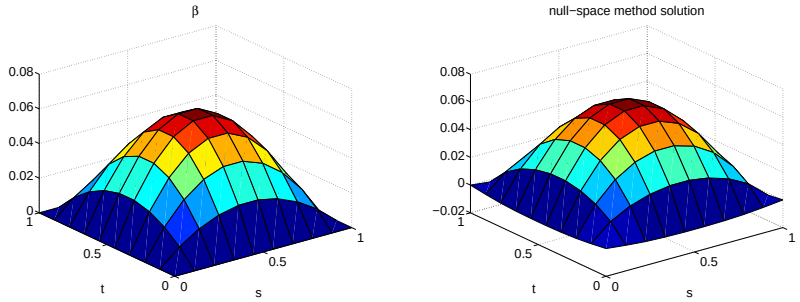

Figure 8.2: $\beta = st(1-s)(1-t)$ and corresponding Poisson solution

where u is the electrostatic potential and f describes the charge density at each point of the domain. Suppose the charge density is given by $f(s,t) = \exp(-(s-1/2)^2 - (t-1/2)^2)$ and that we are interested in the electrostatic potential on the unit square, $\Omega = (0,1) \times (0,1)$. One way to find a unique solution would be to set boundary values on our domain to zero, estimating the boundary as described above. If we solve the problem using finite differences and a zero boundary, we obtain the solution given in Figure 8.1.

Rather than arbitrarily fixing the boundary values, we can instead choose a β function for the null space method whose shape fits our intuitive expectations of the shape of the solution. Or, for a problem that we do not have intuition about, we can set $\beta = 0$ to obtain the minimum-norm least squares solution to the problem. The charge density function for this example is largest in the center of our domain and tapers off toward the edges. It follows that the potential over the field might also be large in the center and taper off toward the edges. If we choose a β function with that shape, $\beta = st(1-s)(1-t)$ (shown on the left in Figure 8.2), we obtain a solution (Figure 8.2, right) that is similar to the the one calculated with zero boundary conditions, but without strictly zero values

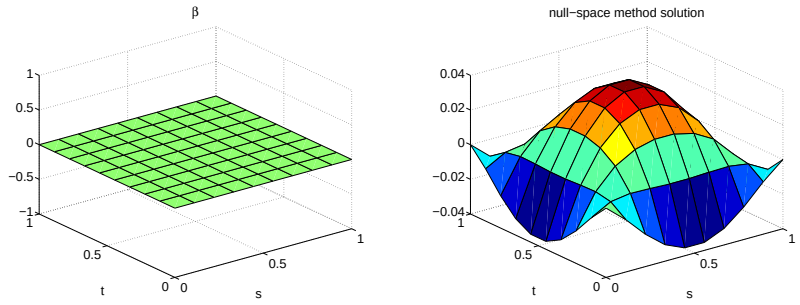

Figure 8.3: $\beta = 0$ and corresponding Poisson solution

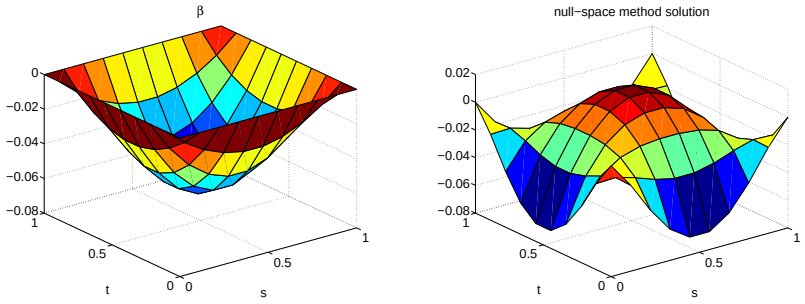

Figure 8.4: $\beta = -st(1-s)(1-t)$ and corresponding Poisson solution

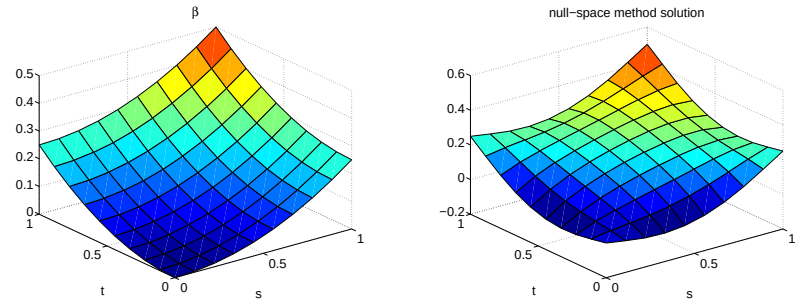

Figure 8.5: $\beta = (s^2 + t^2)/4$ and corresponding Poisson solution

on the boundary.

To illustrate how different choices of β affect the shape of the solution, we include Figures 8.3-8.5, all of which show valid solutions to the given Poisson equation. Figure 8.3 shows the minimum norm least squares solution. These examples are given for a two-dimensional domain. The problem can similarly be solved on a three-dimensional domain, but the results are more difficult to visualize.

8.2 Curl Equation in Materials Science

A second application comes from the field of materials science and specifically considers mechanics of dislocations (lattice defects). That is, the problem equations describe the stress and strain in a metallic solid over time given an initial distribution of dislocations in the material. The part of the problem that we have focused on solving involves the curl of a two-dimensional tensor. As an introduction to the curl operator, we consider the curl applied to a one-dimensional tensor as described by Schey in [46, p. 85-89]. Schey provides the velocity field of a volume of water and uses that to calculate the value of the curl. We will solve the problem backwards, solving

$$\operatorname{curl} \boldsymbol{u} = \boldsymbol{\alpha}, \tag{8.2}$$

where $\boldsymbol{\alpha}$ is selected to be the value of the curl given in the reference. Specifically,

$$\boldsymbol{\alpha} = \begin{bmatrix} 0 \\ 0 \\ 2\omega \end{bmatrix}.$$

For our null-space constraint, we will use

$$\boldsymbol{\beta} = \begin{bmatrix} -\omega s \\ \omega r \\ 0 \end{bmatrix},$$

which is the velocity field Schey gave. In physical terms, ω is the constant angular velocity of the water and we set $\omega = 2$ for our calculations. Since the null-space constraint functions $\boldsymbol{\beta}$ satisfy the curl equation (8.2), our solution should match $\boldsymbol{\beta}$.

Using the unit cube as our domain and a uniform finite element discretization with 9 elements in each dimension, we obtain the results plotted in Figures 8.6-8.9. For each component of the solution, we provide a plot of the entire 3-D volume. For the last component, we also include a cross-section of the rs-plane. We used the program Rocketeer for the 3-D volume plots [22]. The orientation of the axes are described by the figure in the bottom, left corner, and variables are correlated to colors as r - red, s - green, and t - blue. The cross-section is

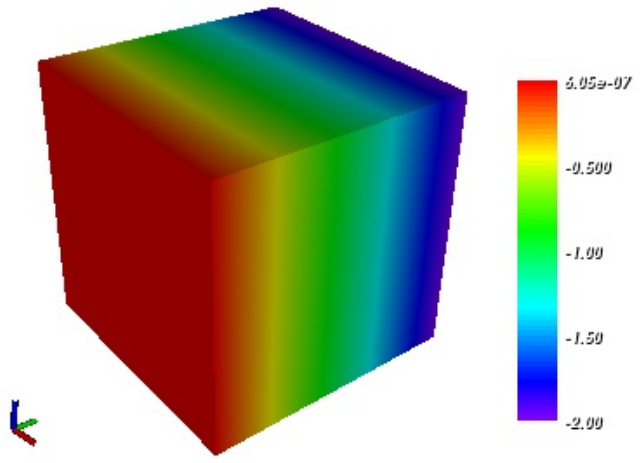

Figure 8.6: Curl example, u_1

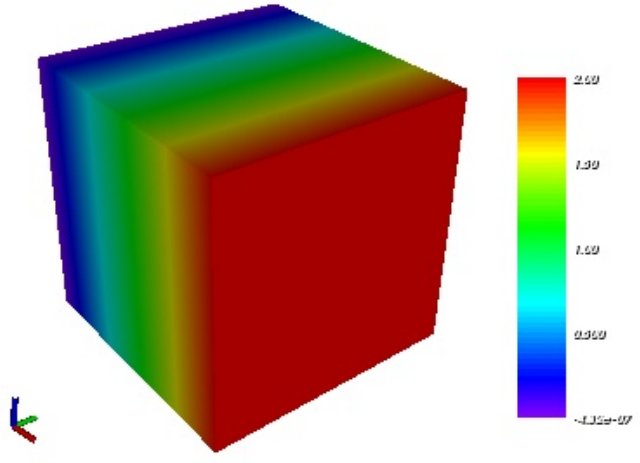

Figure 8.7: Curl example, u_2

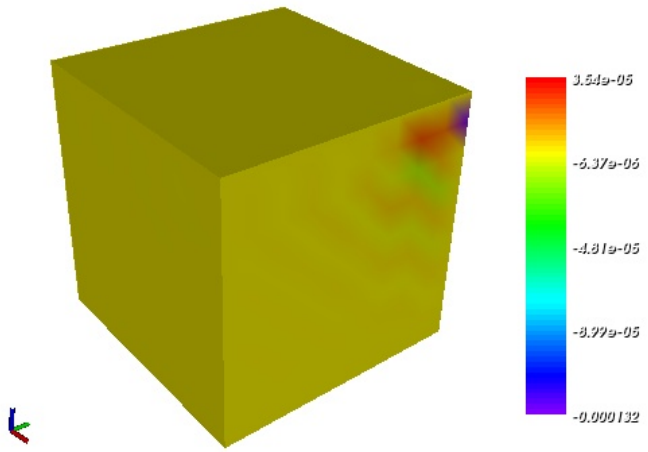

Figure 8.8: Curl example, u_3

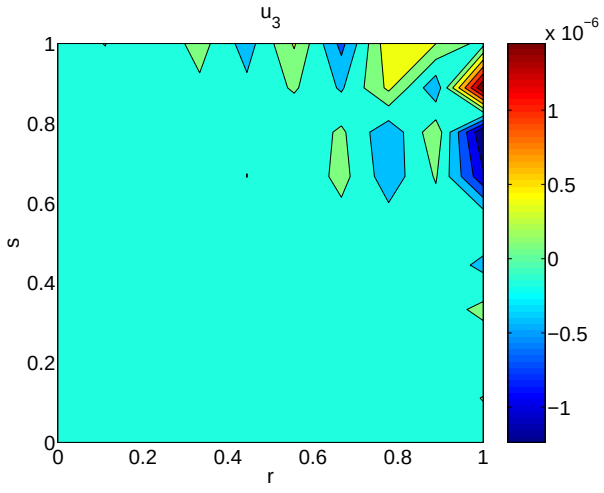

Figure 8.9: Curl example, cross-section of u_3

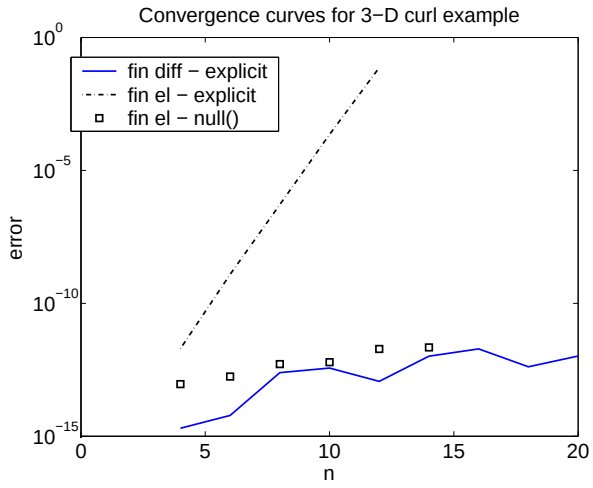

Figure 8.10: Error in approximate solutions for linear curl problem

given at $t = 1/3$. While the "zero" values of the solution (that is, the values of $\boldsymbol{u}_3$) are not exactly zero, the max norm errors for all three parts of our approximate solution compared to the discretized solution functions are similar in magnitude. Specifically, the max norm errors for the three parts are

$$\begin{aligned} \boldsymbol{u}_1\text{:} \quad & 2.8852 \times 10^{-5}, \\ \boldsymbol{u}_2\text{:} \quad & 1.2672 \times 10^{-4}, \\ \boldsymbol{u}_3\text{:} \quad & 1.3169 \times 10^{-4}. \end{aligned}$$

In order to compare errors for various methods as the grid is refined, we solved the problem for several values of n. The results for error, using the max norm of the error of all the components, are plotted in Figure 8.10. While we obtain the correct answer for small values of n, the solution degrades as we use a finer mesh. In fact, the finite element method using an explicit basis becomes so ill-conditioned it appears to have less than full rank for $n \geq 14$, and we omit those invalid data points from the graph. This is the opposite of what we would expect in terms of convergence, unfortunately. These trends of poor convergence correlate with the null residual curve of the explicit null basis for finite differences in Figure 6.14 and with the conditioning curve of the explicit null basis in Figure 6.15, respectively. In both the cases of using explicit bases, we see a loss of accuracy in the problem solution due to loss of accuracy in the null basis in one way or another. It is unclear why the finite element method using the `null()` method in MATLAB loses accuracy, but this loss of accuracy may be an issue for larger problems, even showing up for larger values of n in this example.

As another example, consider a similar problem for which the solution cannot be exactly represented as a linear function. The domain and discretization will

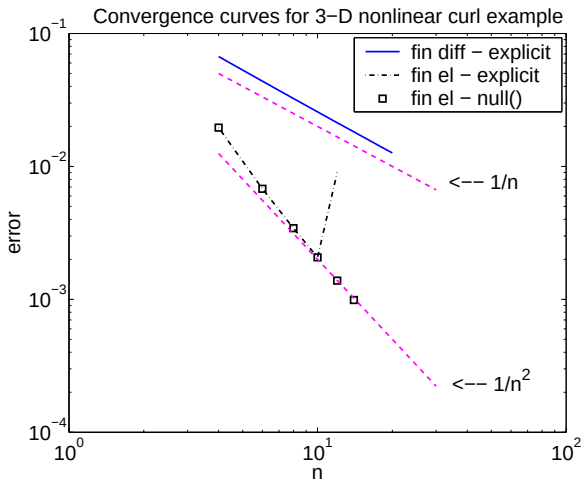

Figure 8.11: Error in approximate solutions for nonlinear curl problem

be the same as for the previous problem, and

$$\boldsymbol{\alpha} = \begin{bmatrix} 0 \\ 0 \\ -2r\exp(-r^2) \end{bmatrix},$$

$$\boldsymbol{\beta} = \begin{bmatrix} 0 \\ exp(-r^2) \\ 0 \end{bmatrix}.$$

This example also comes from Schey [46, p. 88-89]. The vector-field solution u describes an uneven flow in the s direction. As in the prior example, the solution is given by the β function, and here we omit solution plots. Error values for various grid sizes are plotted in Figure 8.11. In this example, they demonstrate the expected convergence as the mesh is refined. Even though the null basis error increases with a finer mesh, the overall error of the system decreases for these values of n because the null basis error is hidden by the larger error of representing a nonlinear function with linear basis functions. For $n = 14$ and higher, however, the null basis error does cause trouble for the finite element, explicit basis method; we omitted data produced by those low rank systems.

The curl operator in the examples above maps a space of one-dimensional tensors (i.e., vectors) to another. In the materials science application we cover next, the curl operator maps two-dimensional 3×3 tensors. The matrix curl operator works in the expected way; the curl of a 3×3 tensor is the curl of each of its rows. The curl equation does not appear by itself in this broader context of dislocations. Rather, the following set of equations is used for those calculations, where all of the tensors are 3×3 tensors.

$$\operatorname{curl} \boldsymbol{u}^p \quad = \quad -\boldsymbol{\alpha} \tag{8.3}$$

$$\boldsymbol{u}^p_{//} = \tilde{\boldsymbol{u}}^p_{//} \tag{8.4}$$

$$\boldsymbol{T} = \boldsymbol{C}\left(\boldsymbol{\epsilon} - \boldsymbol{\epsilon}^p\right);\ \boldsymbol{\epsilon} := \frac{1}{2}\left(\boldsymbol{u} + \boldsymbol{u}^T\right);\ \boldsymbol{\epsilon}^p := \frac{1}{2}\left(\boldsymbol{u}^p + \boldsymbol{u}^{pT}\right) \tag{8.5}$$

$$\operatorname{div}\boldsymbol{T} = \boldsymbol{0} \tag{8.6}$$

$$\dot{\boldsymbol{\alpha}} = -\operatorname{curl}\left(\boldsymbol{\alpha} \times \boldsymbol{V}\right) + \boldsymbol{s} \tag{8.7}$$

$$\dot{\tilde{\boldsymbol{u}}}^p = \boldsymbol{\alpha} \times \boldsymbol{V} \tag{8.8}$$

The variables stand for

- $\boldsymbol{u}^p$ - plastic distortion tensor
- $\tilde{\boldsymbol{u}}^p$ - slip distortion tensor
- $\boldsymbol{\alpha}$ - dislocation density tensor
- $\boldsymbol{u}$ - displacement gradient (strain tensor)
  - $\boldsymbol{\epsilon}$ - symmetric part of $\boldsymbol{u}$
  - $\boldsymbol{\epsilon}^p$ - symmetric part of $\boldsymbol{u}^p$
- $\boldsymbol{C}$ - elastic constitutive tensor
- $\boldsymbol{T}$ - symmetric stress tensor
- $\boldsymbol{V}$ - dislocation velocity vector
- $\boldsymbol{s}$ - dislocation nucleation rate tensor
- $\dot{}$ - time derivative
- // - component of argument field in the null space of the curl

The equations are solved in pairs, and the first pair, Equations 8.3 and 8.4, are the two we wish to solve. The solution $\boldsymbol{u}^p$ that we seek is a 3×3 tensor of functions, and it does not have meaning apart from the following set of equations, 8.5 and 8.6, which give a value for the strain of the system. From the strain, we can calculate the stress tensor, denoted by $\boldsymbol{\sigma}$. The last set of equations update the dislocation density tensor $\boldsymbol{\alpha}$ and thereby simulate the effect of stress and strain through time. More information on mechanics and dislocations can be obtained from [33, 36], and some recent work on theories of dislocations are described in [1, 28]. Understanding the effects of dislocations on stress and strain in metallic solids is important for the design and manufacturing of metals in many arenas, from large-scale components such as bridge girders to small-scale components such as thin films in nano-devices [2].

The pair of equations we are solving correlates exactly with the null-space approach described in Chapter 4, providing a useful application for the research we have done. The null space constraint has not always been used in formulating this problem; the idea of solving the curl problem this way was proposed by Acharya in [2]. Other researchers (see [1, 44, 56, 58], for example) have used

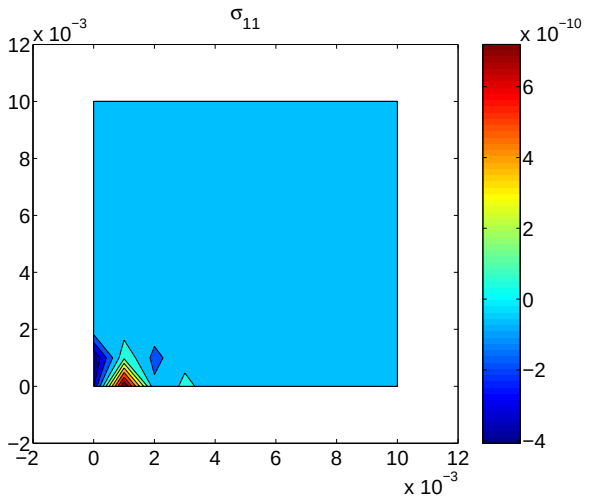

Figure 8.12: Stress of zero stress everywhere, dislocation example

various boundary conditions, seeking a solution for Equations 8.3 and 8.4 that works well for the subsequent calculations. Specifying boundary conditions tends to cause steep gradients in the stress field, however, which are not faithful to the physical problem. The null-space approach offers relief from that issue at least in one case and presumably in others as well.

For demonstrative purposes, we give a couple of examples with known solutions. Our domain for both examples is the cube of length 0.01 lying completely in the first quadrant (i.e., $\Omega = (0, 0.01) \times (0, 0.01) \times (0, 0.01)$), and for our first example we set

$$-\boldsymbol{\alpha} = \begin{bmatrix} 0 & 0 & \alpha_{13} \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix},$$

where

$$\alpha_{13} = 0.1,$$

the constant function 0.1. The stress in a crystalline metal with a constant distribution of dislocations in α_{13} is known to be zero everywhere (see [28]), yet $\boldsymbol{u}^p$ is nonzero. Specifically, the expected solution for the curl portion of the problem is

$$\boldsymbol{u}^p = \begin{bmatrix} -0.05s & 0.05r & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}.$$

We choose this as our $\tilde{\boldsymbol{u}}^p$ function and solve the problem using the null-space approach.

We obtained the results given here using a uniform finite difference discretization with 11 nodes in each spatial dimension. One layer of the solution for the σ_{11} stress component is shown in Figure 8.12, and layers of $\boldsymbol{u}^p_{11}$ and $\boldsymbol{u}^p_{12}$

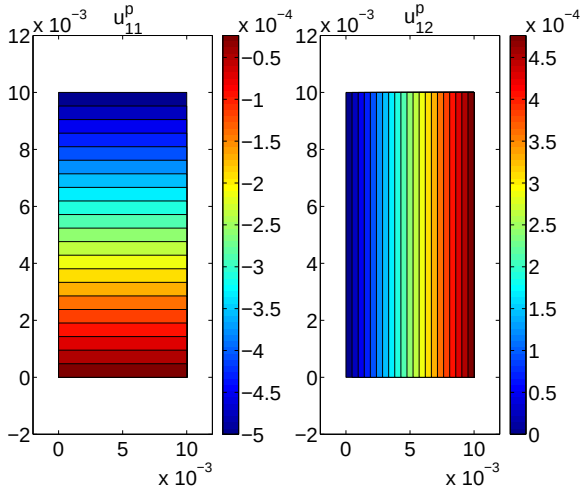

Figure 8.13: $\boldsymbol{u}^p$ data of zero stress everywhere, dislocation example

are given in Figure 8.13. As Figure 8.12 shows, the stress does have a small nonzero portion in one corner, which is undesirable since the stress should be zero everywhere. The nonzero values are quite small, however, and they appear in only one corner. The values of $\boldsymbol{u}^p$ are exactly what we desire, producing the output of $\boldsymbol{u}^p_{11} = -0.05s$ and $\boldsymbol{u}^p_{12} = 0.05r$, as Figure 8.13 shows, and returning essentially zero for all other values of $\boldsymbol{u}^p$, with a maximum nonzero magnitude of 2.35×10^{-16}. In understanding the plots, it is important to recall that the domain of our simulation is the cube of length 0.01, producing an output range of magnitudes 0 to 0.0005. Thus, the plots show the expected results.

For our second example, we demonstrate the null space method applied to a single screw dislocation. The distribution density vector is not meant to handle single dislocations (more discussion of this below), yet this example gives information about whether our method accurately calculates the stress of a problem with a known solution on most of the domain. Along the dislocation the solution has a singularity, and we are not concerned with matching any particular value in that region. In fact, we omit the middle point of the domain when we calculate the error for this problem. If we were concerned about the value of the solution at the singularity we could flatten out the singularity using the finite element method with a weighted inner product that cancels out the singularity in the solution.

For this example, then, we set

$$\boldsymbol{\alpha} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & \alpha_{33} \end{bmatrix},$$

where

$$\alpha_{33} = \begin{cases} 0.1, & r = 0.005,\ s = 0.005,\ 0 \le t \le 1 \\ 0, & \text{elsewhere} \end{cases}.$$

The dislocation distribution corresponds to a single dislocation extending the length of the domain in the t-direction and cutting through the center of rs plane. In lay terms, given a cubic domain with no applied stresses, we can describe such a dislocation as being created by a series of three steps defining a cut and weld operation [28]. The first step is a cut, which is a slice through the domain along the half plane $s = 0$, $0 \le r < 0.5$. The intermediate step is a shift of the cut surfaces some distance (d) with respect to each other, and the last step is a weld that secures the cut surfaces together in new positions. Such a dislocation has a Burger's vector of $(0, 0, d)$.

Relating the value of α_{33} to the Burger's vector of a single dislocation is not entirely straightforward since the dislocation density tensor generally specifies the density of a set of dislocations throughout the domain rather than a single dislocation. According to [1], however, the following relationship holds,

$$\frac{d}{2\pi} = \int_0^{\rho_0} \phi(\rho) \cdot \rho \; d\rho,$$

where ρ is a polar coordinate of radius from the location of the dislocation, or in this case, the center of the rs plane, and ϕ is a positive, scalar-valued function of one variable that vanishes on and outside the cylinder of radius ρ_0. In our example, $\phi(\rho) = 0.1$. We do not have a specific value for ρ_0, though the discretized version of α_{33} has zero values at every point except the center one, implying that $\rho_0 \le \Delta r = \Delta s$. Our sample data includes problems of size $n = 21$, requiring $\rho_0 \le 0.01/21 \approx 4.76 \times 10^{-4}$. We suppose $\rho_0 = 2 \times 10^{-4}$, knowing this is an approximation but giving us something concrete against which to compare our results and to asses convergence. Using this value for ρ_0, we find

$$\begin{aligned} d &= 0.1 \cdot \pi \cdot (p_0)^2 \\ &\approx 1.257 \times 10^{-8}. \end{aligned}$$

We divide by this value below, effectively removing the specific choice of d from the plotted results. Nevertheless, the explanation above is important to our calculation, to enable others to reproduce the results.

Now we are prepared to compare our stress results to the analytical result. The stress caused by a screw dislocation at the center of our domain is

$$\boldsymbol{\sigma} = \frac{\mu d}{2\pi} \begin{bmatrix} 0 & 0 & \frac{-(s-c)}{(r-c)^2+(s-c)^2} \\ 0 & 0 & \frac{r-c}{(r-c)^2+(s-c)^2} \\ \frac{-(s-c)}{(r-c)^2+(s-c)^2} & \frac{r-c}{(r-c)^2+(s-c)^2} & 0 \end{bmatrix},$$

where c is 0.005, the midpoint of $(0, 0.1)$, and μ is the shear modulus determined by the properties of the particular metal we are simulating. Rather than specify

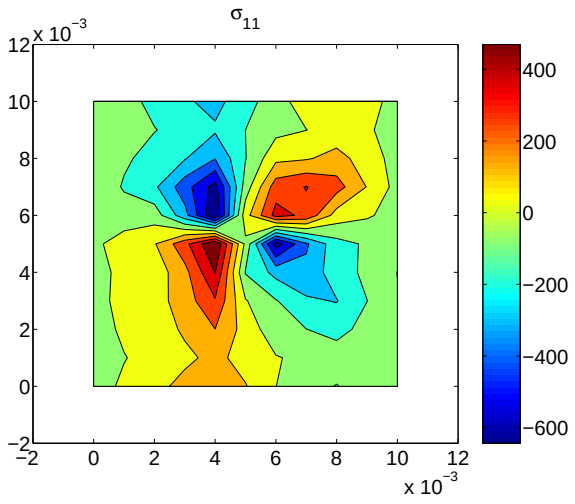

Figure 8.14: $\boldsymbol{\sigma}_{11}$ layer for single dislocation example

a particular μ, we will normalize our stress results by μ. We also normalize our stress results by d so that our expected solution is the right-hand-side of

$$\frac{\boldsymbol{\sigma}}{\mu d} = \frac{1}{2\pi}\begin{bmatrix} 0 & 0 & \frac{-(s-c)}{(r-c)^2+(s-c)^2} \\ 0 & 0 & \frac{r-c}{(r-c)^2+(s-c)^2} \\ \frac{-(s-c)}{(r-c)^2+(s-c)^2} & \frac{r-c}{(r-c)^2+(s-c)^2} & 0 \end{bmatrix}.$$

We applied the null-space approach to this problem using forward finite differencing on a uniform discretization with the $\tilde{\boldsymbol{u}}^p$ tensor set to zero. For $n = 11$, we provide layers of some results in Figures 8.14-8.15, and the expected result for the corresponding layer of σ_{13} in Figure 8.16. For the latter figure, we plotted the expected result discretized with the same number of nodes as the approximation, and we set the middle value to zero in the center of the domain (where the solution is discontinuous) so that the contours would be clearer. The error is quite large for this simulation, but we do see convergence of the approximation to the expected result as the grid is refined. To demonstrate convergence, we plotted our results for the line through $r = 1/2$, $t = 1/2$ for $n = 7$ through $n = 21$ in Figure 8.17 and error values in Figure 8.18. As long as we are producing the correct stress values outside the region of discontinuity, the simulation is valid. The two vertical lines on the plot in Figure 8.17 delineate the region of discontinuity in the middle of the domain. To obtain error values, we calculated the maximum absolute difference between the approximate solution and the expected solution for σ_{13} at each grid point, omitting the center point of of the grid where the solution is discontinuous.

Forward finite differences are off-centered, and that is reflected in the results we obtain from such a discretization. A finite element discretization provides symmetric results, but because of ill-conditioning of the explicit basis and computational intensity of calculating an orthogonal basis, we had difficulties solving

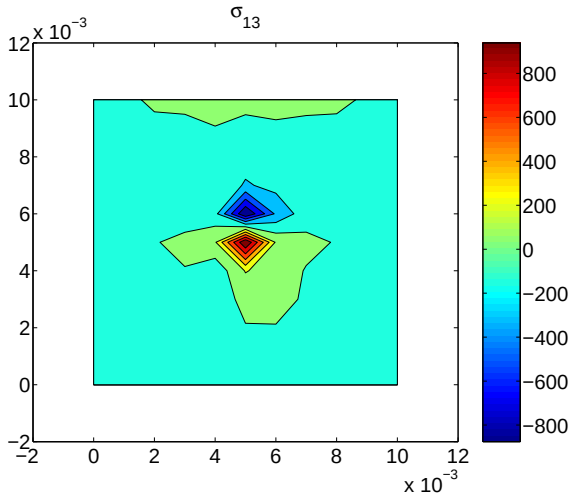

Figure 8.15: $\boldsymbol{\sigma}_{13}$ layer for single dislocation example

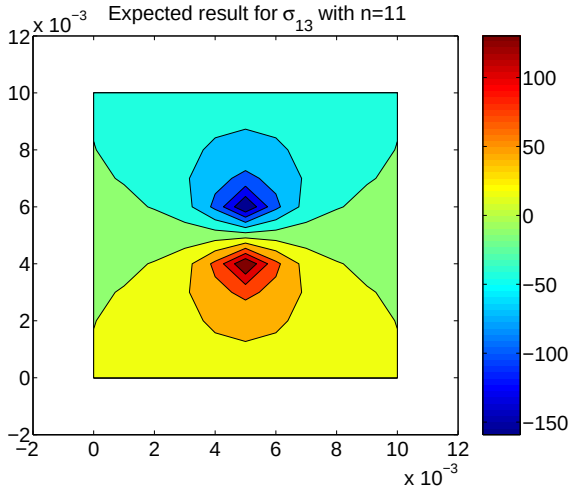

Figure 8.16: Expected solution layer for single dislocation example

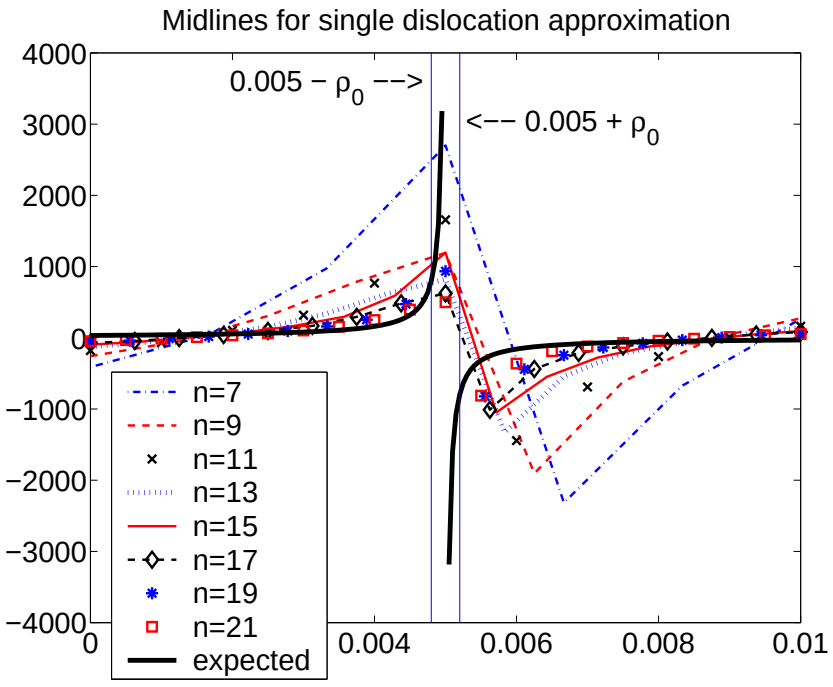

Figure 8.17: Approx. solutions along center line for single dislocation example

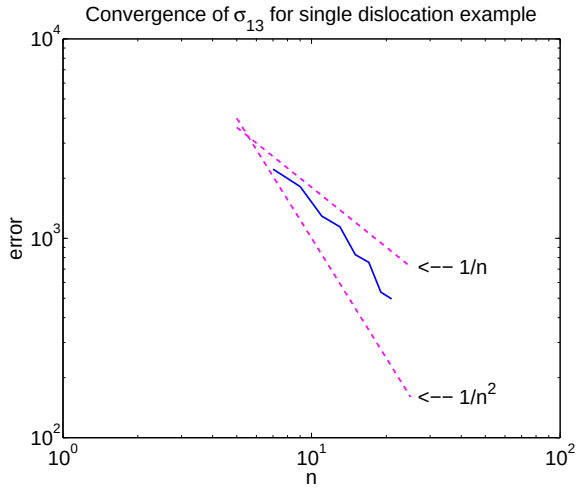

Figure 8.18: Error in σ_{13} approximations for single dislocation example

problems with $n > 15$. Since we have more data for the finite difference method, we chose it for the results to present here. The convergence of the finite difference method was linear for the earlier curl example, as expected from the first-order finite difference approximations used. Here we do even better than linear convergence for reasons not entirely understood but perhaps related to the approximation of ρ required as part of this simulation. More importantly, the results show that our solution is converging to the expected answer, and we expect this trend to continue for larger values of n.

9 Conclusions and Future Work

In this thesis we have presented the null-space approach to solving differential equations. This approach is useful for problems either without boundary conditions or with boundaries at infinity. We have proved that this approach provides a unique solution and have included results from several implementations in two and three dimensions, for finite difference, finite element, and spectral methods. These results demonstrate convergence of null-space methods to expected answers for example problems and for a mechanics problem in materials science.

As implied by their name, null-space methods require knowledge of a null-space. We discussed options for finding a basis for a null-space for a general problem and present explicitly-defined null bases that we derived for four discrete operators on specific domains. (In three cases, we also proved that the null basis is a basis for all problem sizes.) To demonstrate the benefits of the various null-basis options, we compared several characteristics of each, showing that the explicit bases are best except in regard to conditioning. Ill-conditioning of the explicit bases is controlled by heuristics that trade-off sparsity and conditioning.

The problems we solved in this work produce linear systems small enough to be solved by direct methods. We also presented information on iterative solvers. Even if the linear system is well-conditioned, the null-space method produces a matrix with an eigenvalue distribution that leads to poor convergence of iterative methods, especially for the Laplacian operator. We have found that an incomplete LU preconditioner makes iterative methods viable.

In addition to these major topics, we have included additional information throughout the thesis including somewhat novel notation, comparison of boundary conditions and null-space specification of problems with boundary conditions, results relating the continuous and discrete null spaces, and discussion of why orthogonalization does not always maintain accuracy of a null basis.

This work lays the foundation for much additional research. Specifically, the null-space approach to solving differential equations gives additional motivation for finding large, sparse, well-conditioned null bases and for developing code that will be effective for large problems. Many papers discuss algorithms for finding sparse null bases, and the authors must have programmed some implementation for obtaining results for their papers, but as far as we know, these algorithms have not resulted in code that is publicly accessible. The implementation we have been using, which includes the turnback method and its variations, becomes slow for larger problems and faces ill-conditioning as the problem size grows,

leaving room for improvement. Related to the problem of finding a null basis is the task of improving the conditioning of a basis without resorting to an orthogonalization routine and the resulting fill. While we have already addressed that issue in some detail, we believe there may be even better heuristics or a method that creates a null basis that offers a somewhat sparse, well-conditioned null basis, perhaps along the lines of an orthogonalized turnback method. (We have begun some preliminary work on such a method, but this work is premature to present at this point.) Another avenue for research in this area would involve a deeper look at graph algorithms, such as those used in [4, 15].

In addition to improving general methods for finding a null basis, an interesting extension of our work would be to find explicit null bases for additional operators, discretizations, and domains. Our work with explicit bases is limited to square and cubic domains with uniform discretizations. It is possible that these formulas could be generalized to less specific domains. Or, perhaps these formulas could be generalized to tetrahedral elements from the cubic elements used in our application. As we showed in our work, explicit null bases are extremely fast in comparison to general methods for computing null bases; progress in generalizing explicit null bases would speed up null-space methods for a variety of problems, lending additional credence to the use of null-space methods instead of conventional methods with approximated boundary conditions. The use of different elements may assist in this endeavor. Nedelec elements [27, 39], for example, are chosen to include functions that are in the null space of the curl operator.

Finally, while we have laid the foundation for the null-space approach and have proved fundamental theorems regarding the discretized linear system, more questions wait to be answered about the relationship between discrete and continuous null bases. For example, how do the explicit, discrete null vectors correlate to continuous functions in the null space? Can a discretization be chosen so that discretized null functions are null vectors? In higher dimensional problems, which of the infinite number of null functions should be represented by the finite number of null vectors? The answer to this last question may lie in a study of n-widths, which are used in approximation theory [41].

This research explores a new approach to solving differential equations and opens the door for future avenues of research. In many ways this thesis is not a conclusion but is a beginning of other work to come.

A Derivations and Computations

This appendix provides derivations that verify claims in the thesis and that may be of interest to the reader but are not important to following the text.

A.1 Null-Space Portion of 3-D Curl Problem

As explained in Section 6.2.4, the 3-D curl problem matrix is of dimension $3(n-2)^3 \times 3n^3$ and has a null space of dimension $n^3+6n^2+12n-40$. Assuming the extra row dependencies are removed from the original matrix, we simply take the dimension of the null space over the number of columns ($3n^3$) to find what percent of the problem is composed of null-space constraints. Specifically,

$$\frac{n^3+6n^2+12n-40}{3n^3}.$$

For the value $n = 10$, this fraction evaluates to $1680/3000 = 0.56$ or 56 percent. For $n = 100$, the fraction evaluates to $1061160/3000000 \approx 0.35$ or about 35 percent. Using L'Hopital's rule, we can calculate

$$\lim_{n\to\infty} \frac{n^3+6n^2+12n-40}{3n^3} = \frac{1}{3}.$$

Thus, the null-space criterion determines at least one-third of the constraints for the problem.

The result is similar when extra row dependencies in the original matrix are not removed. In this case, the total number of rows is $3(n-2)^3+n^3+6n^2+12n-40 = 4n^3+24n-64$, and in the limit, the null-space constraints provide at least one-fourth of the rows. While the percentage of rows given by the null-space is smaller in this calculation than in the previous case, the percentage of unique constraints that the null-space adds to the linear system is still one-third of the total.

A.2 Number of Operations for Turnback Method

Assume our original problem matrix $\boldsymbol{D}$ is of dimension $m \times p$ with $m < p$. The turnback method begins with a QR factorization of the matrix. That factorization requires on the order of $m^2p - m^3/3$ operations. Then, for each

null vector, turnback computes a small QR factorization of dimension (number of "active rows") by (number of nonzeros in null vector). We will assume all rows are active (m), that all null vectors have approximately k nonzero values with $k < m$, and that the original matrix has full row rank. For each of our $p-m$ null vectors, then, we have a factorization requiring $\mathcal{O}(k^2m - k^3/3)$ operations for a total on the order of $(k^2m - k^3/3)(p-m) = k^2(mp - m^2) - k^3/3(p-m)$. Adding in the original QR factorization, we have a total number of operation of $\mathcal{O}(m^2p - m^3/3 + k^2(mp - m^2) - k^3/3(p-m)$.

The expression above does not tell us much; the added variable just adds complexity to the formula. If we assume additional information about k, we can more easily compare this result with others involving only p and m. In a good scenario, k will be proportional to $\sqrt{p}$. In this case, our number of operations is $\mathcal{O}(mp^2 - 1/3(m^3 + p^{5/2} - mp^{3/2}))$. In the best scenario, k might be proportional to some constant independent of p. Assuming that, we obtain a number of operations of $\mathcal{O}(m^2p - m^3/3 + mp - m^2)$ when we drop lower order terms.

These results do not take into consideration any operations required for variants of the turnback that track and search information about columns at each step of the algorithm, however, those added operations presumably will be balanced out by smaller QR factorizations and thus result in fewer or a similar number of operations overall. That said, these numbers are approximations based on the assumptions we made, and may not be valid for some problems.

A.3 Proof That Alternate Form of Linear System Is Full Rank

In this section we prove that the linear system given in equation (7.2) is nonsingular for discrete sampling and linear combination approximations. For reference, the equation is

$$\begin{bmatrix} \boldsymbol{D}^T\boldsymbol{D} & \hat{\boldsymbol{Z}} \\ \hat{\boldsymbol{Z}}^T & \boldsymbol{0} \end{bmatrix} \begin{bmatrix} \boldsymbol{x} \\ \boldsymbol{y} \end{bmatrix} = \begin{bmatrix} \boldsymbol{D}^T\boldsymbol{a} \\ \boldsymbol{b} \end{bmatrix},$$

with $\boldsymbol{D}$ an $m \times p$ matrix of rank k, $\boldsymbol{Z}$ an $p \times p-k$ matrix that is a null basis for $\boldsymbol{D}$ and $\hat{\boldsymbol{Z}}$ given by $\boldsymbol{\Phi}^*\boldsymbol{\Phi}\boldsymbol{Z}$, where $\boldsymbol{\Phi}$ contains linearly independent basis functions, implying that $\boldsymbol{\Phi}^*\boldsymbol{\Phi}$ is symmetric positive definite.

Lemma 17 *The matrix of the linear system in (7.2), derived from an optimization perspective of the linear system produced by the null-space approach for solving differential equations, is nonsingular for discrete sampling and linear combination approximations.*

Proof: We consider this blocked system as two sets of columns. First, we note that the product $\boldsymbol{D}^T\boldsymbol{D}$ can be considered a set of linear combinations of

the columns of $\boldsymbol{D}^T$ or as a set of linear combinations of the rows of $\boldsymbol{D}$. Further, the product has rank k. (We omit the proof that rank$(\boldsymbol{D}^T\boldsymbol{D}) = k$ which can be shown using SVD's of both matrices.)

Since $\boldsymbol{D}^T\boldsymbol{D}$ is a set of linear combinations of the rows of $\boldsymbol{D}$, $\boldsymbol{Z}$ contains valid null vectors, and the left half of the matrix above,

$$\begin{bmatrix} \boldsymbol{D}^T\boldsymbol{D} \\ \hat{\boldsymbol{Z}}^T \end{bmatrix},$$

has full rank by application of Lemma 3 in Chapter 5. By symmetry, we can also conclude that the top half of the matrix,

$$[\, \boldsymbol{D}^T\boldsymbol{D} \quad \hat{\boldsymbol{Z}} \,],$$

has full row rank of p. Since $\boldsymbol{D}^T\boldsymbol{D}$ has rank k, $\hat{\boldsymbol{Z}}$ has $p-k$ columns that are linearly independent from columns of $\boldsymbol{D}^T\boldsymbol{D}$. Therefore, the right half of our matrix,

$$\begin{bmatrix} \hat{\boldsymbol{Z}} \\ \boldsymbol{0} \end{bmatrix},$$

is linearly independent from the left half and has rank $p-k$. The total rank, then, is $p+p-k = 2p-k$, which is the dimension of our matrix. Thus, the matrix is nonsingular.

A.4 Accuracy After Orthogonalization

In Section 6.3, we consider methods for finding a null basis that include a primary method producing a sparse null basis followed by a subsequent QR factorization. In some cases, the subsequent QR factorization reduced the accuracy of the basis (using $\boldsymbol{Q}$ as the basis). That is, the null residual values degraded as the problem size increased. Investigation has demonstrated that the ill-conditioning of the null basis contributes to, if not entirely causes, this trend. (Since there is some rounding error in the process of factorization, ill-conditioning may not be the only contributing factor. See [31, 57] for more information on rounding error in QR factorization.)

Consider the product $\boldsymbol{F}$ of our original matrix $\boldsymbol{A}$ and our explicit null basis $\boldsymbol{Z}$. We have observed that the null residual is approximately machine precision, ϵ, that is,

$$\|\boldsymbol{F}\| = \|\boldsymbol{A}\boldsymbol{Z}\| = \mathcal{O}(\epsilon)\|\boldsymbol{A}\|\|\boldsymbol{Z}\|.$$

Supposing we have an exact QR factorization of $\boldsymbol{Z}$, we have

$$\|\boldsymbol{F}\| = \|\boldsymbol{A}\boldsymbol{Q}\boldsymbol{R}\| = \mathcal{O}(\epsilon)\|\boldsymbol{A}\|\|\boldsymbol{Q}\|\|\boldsymbol{R}\|.$$

Setting $\boldsymbol{G} = \boldsymbol{AQ} = \boldsymbol{FR}^{-1}$, we have

$$\|\boldsymbol{G}\| \leq \|\boldsymbol{F}\|\|\boldsymbol{R}^{-1}\| \leq \mathcal{O}(\epsilon)\|\boldsymbol{A}\|\|\boldsymbol{Q}\|\kappa(\boldsymbol{R}),$$

where $\kappa(\boldsymbol{R})$ is the condition number of $\boldsymbol{R}$. Ill-conditioning of $\boldsymbol{Z}$ will appear in the $\boldsymbol{R}$ factor of the QR factorization; it follows that the null residual of $\boldsymbol{AQ}$ may be large when the original null basis is ill-conditioned.

References

[1] A. Acharya. A model of crystal plasticity based on the theory of continuously distributed dislocations. *J. Mech. Phys. Solids*, 49:761–784, 2001.

[2] A. Acharya. Dislocation mechanics and homogenized plasticity – applied mathematical research in multiscale crystal defect physics. White paper, 2002.

[3] I. Adler, N. Karmarkar, M. G. C. Resende, and G. Veiga. Data structures and programming techniques for the implementation of Karmarkar's algorithm. *ORSA J. Comput.*, 1:84–106, 1989.

[4] M. Arioli, J. Maryska, M. Rozloznik, and M. Tuma. Dual variable methods for mixed-hybrid finite element approximation of the potential fluid flow problem in porous media. *Electronic Trans. Numer. Anal.*, 22:17–40, 2006.

[5] Z. Battles and L. N. Trefethen. An extension of MATLAB to continuous functions. *SIAM J. Sci. Comput.*, 25:1743–1770, 1994.

[6] M. W. Berry, M. T. Heath, I. Kaneko, M. Lawo, R. J. Plemmons, and R. C. Ward. An algorithm to compute a sparse basis of the null space. *Numer. Math.*, 47:483–504, 1985.

[7] P. Bochev and R. B. Lehoucq. On the finite element solution of the pure Neumann problem. *SIAM Review*, 47:50–66, 2005.

[8] R. F. Boisvert, R. Pozo, K. Remington, R. Barrett, and J. J. Dongarra. The Matrix Market: A web resource for test matrix collections. In Ronald F. Boisvert, editor, *Quality of Numerical Software, Assessment and Enhancement*, pages 125–137, London, 1997. Chapman & Hall.

[9] R. A. Brualdi, S. Friedland, and A. Pothen. The sparse basis problem and multilinear algebra. *SIAM J. Matrix Anal. Appl.*, 16:1–20, 1995.

[10] G. F. Carrier and C. E. Pearson. *Partial Differential Equations: Theory and Technique*. Academic Press, San Diego, CA, 1976.

[11] S. F. Chang and S. T. McCormick. A hierarchical algorithm for making sparse matrices sparser. *Math. Prog.*, 56:1–30, 1992.

[12] S. F. Chang and S. T. McCormick. Implementation and computational results for the hierarchical algorithm for making sparse matrices sparser. *ACM Trans. Math. Soft.*, 19(3):419–441, 1993.

[13] K. Chen. *Matrix Preconditioning Techniques and Applications*. Cambridge University Press, Cambridge, UK, 2005.

[14] T. F. Coleman and A. Pothen. The null space problem I: Complexity. *SIAM J. Algebraic Discrete Meth.*, 7:527–537, 1986.

[15] T. F. Coleman and A. Pothen. The null space problem II: Algorithms. *SIAM J. Algebraic Discrete Meth.*, 8:544–563, 1987.

[16] J. D. F. Cosgrove, J. C. Diaz, and A. Griewank. Approximate inverse preconditioning for sparse linear systems. *Int. J. Comput. Math.*, 44:91–110, 1992.

[17] T. A. Davis. *Direct Methods for Sparse Linear Systems*. SIAM, Philadelphia, PA, 2006.

[18] C. de Boor. An alternative approach to (the teaching of) rank, basis, and dimension. *Linear Algebra Appl.*, 146:221–229, 1991.

[19] I. S. Duff, A. M. Erisman, and J. K. Reid. *Direct Methods for Sparse Matrices*. Oxford University Press, New York, 1986.

[20] T. A. Elkins. Orthogonal harmonic functions in space. *Proc. Amer. Math. Soc.*, 8:500–509, 1957.

[21] A.S. Farooqui. A complete set of orthonormal harmonic functions. *SIAM J. Math. Anal.*, 4:309–313, 1973.

[22] R. A. Fiedler and J. C. Norris. Rocketeer User's Guide, 2005. http://www.csar.uiuc.edu/rocstar/rocketeer/.

[23] A. George and J. W.-H. Liu. *Computer Solution of Large Sparse Positive Definite Systems*. Prentice Hall, Englewood Cliffs, NJ, 1981.

[24] J. R. Gilbert and M. T. Heath. Computing a sparse basis for the null space. *SIAM J. Algebraic Discrete Meth.*, 8:446–459, 1987.

[25] J. Gondzio. Presolve analysis of linear programs prior to applying an interior point method. *INFORMS J. Comput.*, 9:73–91, 1997.

[26] N. I. M. Gould and J. A. Scott. Sparse approximate-inverse preconditioners using norm-minimization techniques. *SIAM J. Sci. Comput.*, 19:605–625, 1998.

[27] M. Hano, H. Komatsu, and K. Taniguchi. Systematic construction of three-dimensional ultra high order Nedelec's elements. *IEEE Trans. on Magnetics*, 36:1623–1626, 2000.

[28] A. K. Head, S. D. Howison, J. R. Ockendon, and S. P. Tighe. An equilibrium theory of dislocation continua. *SIAM Review*, 35:580–609, 1993.

[29] M. T. Heath. *Scientific Computing: An Introductory Survey*. McGraw-Hill, New York, 2nd edition, 2002.

[30] M. T. Heath, R. J. Plemmons, and R. C. Ward. Sparse orthogonal schemes for structural optimization using the force method. *SIAM J. Sci. Stat. Comput.*, 5:514–532, 1984.

[31] N. J. Higham. *Accuracy and Stability of Numerical Algorithms*. SIAM, Philadelphia, PA, 1996.

[32] A. J. Hoffman and S. T. McCormick. A fast algorithm that makes matrices optimally sparse. In *Progress in Combinatorial Optimization*, pages 185–196. Academic Press, New York, 1984. W. R. Pulleyblank, ed.

[33] D. Hull and D. J. Bacon. *Introduction to Dislocations*. Pergamom Press, Elmsford, NY, 3rd edition, 1984.

[34] I. Kaneko, M. Lawo, and G. Thierauf. On computational procedures for the force method. *Internat. J. Numer. Meth. Engrg.*, 18:1469–1495, 1982.

[35] C. Lanczos. *Linear Differential Operators*. Dover, New York, 1997. (Reprint of 1961 edition).

[36] J. Lemaitre and J.-L. Chaboche. *Mechanics of Solid Materials*. Cambridge University Press, Cambridge, UK, 2000. (Translated from French by B. Shrivasta), (Reprint of 1990 edition).

[37] S. T. McCormick. Making sparse matrices sparser: Computational results. *Math. Prog.*, 49:91–111, 1990.

[38] J. A. Meijerink and H. A. Van Der Vorst. An iterative solution method for linear systems. *Math. Comp.*, 31:148–162, 1977.

[39] J. C. Nedelec. Mixed finite elements in $\mathbb{R}^3$. *Numer. Math.*, 35:315–341, 1980.

[40] E. Pergola and R. A. Sulanke. Schröder triangles, paths, and parallelogram polyominoes. *J. Integer Sequences*, 1, 1998. Article 98.1.7.

[41] A. Pinkus. *n-Widths in Approximation Theory*. Springer-Verlag, Berlin, Germany, 1985.

[42] R. J. Plemmons and R. E. White. Substructuring methods for computing the nullspace of equilibrium matrices. *SIAM J. Matrix Anal. Appl.*, 11:1–22, 1990.

[43] A. Pothen. Sparse null basis computations in structural optimization. *Numer. Math.*, 55:501–519, 1989.

[44] A. Roy and A. Acharya. Finite element approximation of field dislocation mechanics. *J. Mech. Phys. Solids*, 53:143–170, 2005.

[45] Y. Saad and M. H. Schultz. GMRES: a generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM J. Sci. Comput.*, 7:856–869, 1986.

[46] H. M. Schey. *div, grad, curl and all that*. W. W. Norton & Company, New York, 3rd edition, 1997.

[47] C. Siefert and E. de Sturler. Preconditioners for generalized saddle-point problems. *SIAM J. Numer. Anal.*, 44:1275–1296, 2006.

[48] N. J. A. Sloane. The on-line encyclopedia of integer sequences, 2005. http://www.research.att.com/∼njas/sequences/, see sequences A033877, A006318, A072335.

[49] J. Stern and S. Vavasis. Nested dissection for sparse nullspace bases. *SIAM J. Matrix Anal. Appl.*, 14:766–775, 1993.

[50] G. W. Stewart. *Afternotes Goes to Graduate School: Lectures on Advanced Numerical Analysis*. SIAM, Philadelphia, PA, 1998.

[51] G. W. Stewart. Error analysis of the quasi-Gram-Schmidt algorithm. 2004. Institute for Advanced Computer Studies, University of Maryland, TR-2004-17.

[52] A. Topcu. *A contribution to the systematic analysis of finite element structures through the force method*. PhD thesis, University of Essen, Germany, 1979. (in German).

[53] L. N. Trefethen and D. Bau III. *Numerical Linear Algebra*. SIAM, Philadelphia, PA, 1997.

[54] A. Tveito and R. Winther. *Introduction to Partial Differential Equations: A Computational Approach*. Springer-Verlag, New York, 1998.

[55] H. A. van der Vorst. Bi-CGSTAB: a fast and smoothly converging variant of Bi-CG for the solution of nonsymmetric linear systems. *SIAM J. Sci. Comput.*, 13:631–644, 1992.

[56] S. N. Varadhan, A. J. Beaudoin, A. Acharya, and C. Fressengeas. Dislocation transport using an explicit Galerkin/least-squares formulation. *Modelling Simul. Mater. Sci. Eng.*, 14:1245–1270, 2006.

[57] J. H. Wilkinson. *The Algebraic Eigenvalue Problem*. Clarendon Press, London, England, 1965.

[58] J. R. Willis. Second-order effects of dislocations in anisotropic crystals. *Int. J. Engng. Sci.*, 5:171–190, 1967.

[59] X. Ye and C. A. Hall. The construction of a null basis for a discrete divergence operator. *J. Comput. Appl. Math.*, 58:117–133, 1995.

Author's Biography

Hanna Joy Neradt was born and raised in Illinois, the sixth of seven children of Richard and Albertha Vander Zee. Except for the time she spent at George Washington University in 1999 for the Summer Program for Women in Mathematics, Hanna has lived exclusively in Illinois, achieving a B.A. in mathematics and computer science in 2000 at Trinity Christian College in Palos Heights and subsequently pursuing doctoral work in computer science at the University of Illinois at Urbana-Champaign (UIUC). Her final year at Trinity, Hanna was chosen to be the student laureate to the Lincoln Academy for her class, and at UIUC she was named both an Illiac Fellow and a SURGE (Scholarship for Under-Represented Groups in Engineering) Fellow. Hanna recently married Brian Neradt and plans to take time to be a mom after finishing her doctoral degree.